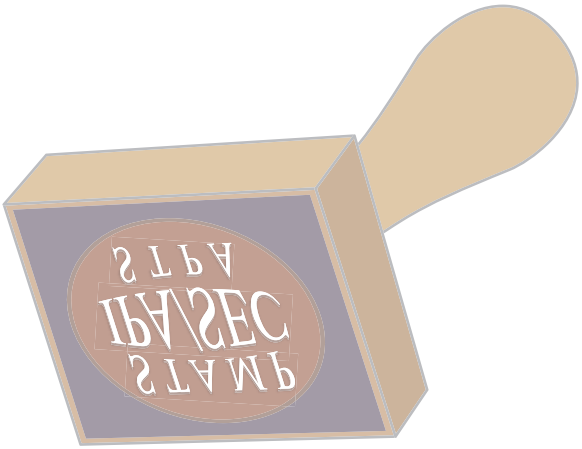
IPA Better Life with IT

はじめての STAMP/STPA

～システム思考に基づく新しい安全性解析手法～

独立行政法人情報処理推進機構
Information-technology Promotion Agency, Japan（IPA）
技術本部 ソフトウェア高信頼化センター
Software Reliability Enhancement Center（SEC）
ソフトウェア高信頼化推進委員会
Software Reliability Enhancement Promotion Committee
システム安全性解析手法 WG
System Safety Analysis WG

Ver.1.0
2016年3月



STAMP
IPA/SEC
STPA

はじめに

本書は、独立行政法人情報処理推進機構（IPA）技術本部ソフトウェア高信頼化センター（SEC）に 2015 年度に新設したシステム安全性解析手法 WG の初年度の活動に基づいてまとめた報告書である。

近年、システムが大規模・複雑になり、さらにネットワークによって相互に接続されて、システム障害もシステム構成要素に起因するのみならず、構成要素同士の間、ないし、システムと人間との間の複雑な相互作用によるものがしばしば発生している。かかる状況において、本 WG では、システムの安全性に関して世界的に著名なマサチューセッツ工科大学（MIT）の Nancy G. Leveson 教授が提唱している STAMP/STPA (Systems Theoretic Accident Model and Processes / System-Theoretic Process Analysis) に着目して、その理解と有効性の確認、適用事例研究の実践などを当面の主な目的として活動することとした。本 WG には Leveson 教授のグループと長く交流して STAMP/STPA の知識と経験を有する委員やシステムの安全性とその分析への STAMP/STPA 適用に興味を持つ委員などが参加している。

2015 年 6 月に SEC 特別セミナーとして開催した「システムベースのエンジニアリング最新動向：複雑化するシステムの安全性とセキュリティを確保するためにすべきこと !」において Leveson 教授に講演をして頂き、その翌日には、本 WG の委員ならびに関係者と Leveson 教授との意見交換会を持つことができた。この意見交換会によって、設置したばかりの本 WG の活動に弾みがついて、WG 委員から具体的な対象に対する専門領域の知識の提供を受けて STAMP/STPA の適用事例研究を WG 全体で共有するかたちで進めることとなった。

2016 年 1 月には、国立研究開発法人宇宙航空研究開発機構（JAXA）と IPA との共催で開催した第 13 回クリティカルソフトウェアワークショップ（13th WOCS[2]）においても Leveson 教授に特別基調講演をお願いし、それに引き続いて行った Leveson 教授ならびに John Thomas 博士と、本 WG 委員および関係者との意見交換会において本 WG の活動概要を報告した。その際に、特に、上述の事例研究に対するコメントを両氏から頂戴したところ、初歩的な例ではあるけれども、対象システムのモデル化ならびに安全性分析方法について、良好な評価を受けた。本 WG としては、まずは、STAMP/STPA を適切に理解しているとの自負を得たところである。

本書の内容は、目次に示す通りである。第 1 章と第 2 章には、我が国における STAMP/STPA のエキスパートによる解説をつけた。STAMP/STPA の初学者にとってわかり易い入門解説として有用であろう。本 WG で行った適用事例研究の具体的対象として鉄道の踏切制御装置について第 3 章で述べ、適用事例研究の結果を第 4 章に示す。この適用事例研究は、STAMP/STPA 初学者にとって、本 WG における入門的適用経験を追体験して共有することにより、STAMP/STPA の理解と修得に有益なものになっていると期待している。第 5 章には、支援ツールの活用、ならびに、従来のシステム記述や分析によるシステムの振舞いや特性の理解に基づいて STAMP/STPA を適用する試みについて述べる。第 6 章では、エンタープライズ系のシステムに対する STAMP 適用についての検討を行う。

本書が、システムの安全性に関するモデル化および分析方法に関して、いくばくなりとも有用な情報を提供するものとなれば幸いである。

目次

図表目次

1. STAMP 解説

1.1. アクシデントモデル STAMP

20 世紀に開発されたシステムの多くは、構成要素が少なく、それらの役割も明確であり、それゆえアクシデントが起きた場合は、構成要素のどれが根本原因であり、アクシデントに至ったのかを分析することは容易であった。従来アクシデントは、根本原因となる機器の故障や人間のオペレーションミスがまず発生し、それがシステム内の他の機器や人間に伝搬し、システム内で悪影響を食い止めることができずに、最終的に起こってしまうという、チェーンオブイベントの形のモデルとして捉えられていた。

しかし、21 世紀になると、開発されるシステムの規模は非常に大きくなり、構成要素も爆発的に増大するとともに、要素間の相互作用も複雑になり、個々の要素の役割を理解しただけでは、もはやシステムを理解できなくなった。そのような相互作用が複雑なシステムにおけるアクシデントの原因は、一つの構成要素に限定できる要因だけではなく、複数の要素間の相互作用による要因も考える必要があった。

マサチューセッツ工科大学の Leveson 教授は、著書「Engineering a Safer World」[Leveson2012] の中で、システムの安全性は構成要素の相互作用から創発されるものであり、個々の要素を分割して分析するべきではないと述べた。そして、現代のシステムのアクシデントの多くは、システム構成要素の故障によって起きるのではなく、システムの中で安全のための制御を行う要素（コントローラー：Controller）と制御される要素（被コントロールプロセス：Controlled Process）の相互作用が働かないことによって起きるというアクシデントモデルを提唱した。このモデルを「STAMP（Systems-Theoretic Accident Model and Process）：システム理論に基づくアクシデントモデル」と呼ぶ。

STAMP モデルでは、システムの様々な階層でコントローラーと被コントロールプロセスに該当する要素が存在しており、それらの相互作用が適切に働くことによりシステムの安全が実現されるとする。STAMP モデルにおいて、アクシデントは相互作用が適切に働かないことによって起こり、具体的にはコントローラーから被コントロールプロセスへの必要な制御指示（コントロールアクション：Control Action という。以下、コントロールアクション）が適切に与えられないために起こるとしている。そして、不適切なコントロールアクションが与えられる要因として、コントローラー自身が想定する被コントロールプロセスの状態（これをプロセスモデル：Process Model という）が、実際の被コントロールプロセスの状態を正しく反映できていないことが主要な要因であるとしている。たとえコントローラーも被コントロールプロセスも故障せずに、仕様通りに正しく動作していても、このような認識の不整合により不適切なコントロールアクションが与えられ、最終的にアクシデントにつながるというアクシデントモデルなのである。

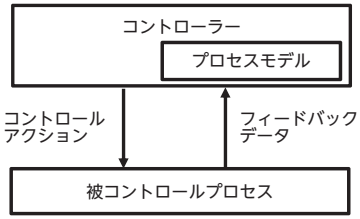


図 1.1-1　STAMP における相互作用のモデル

1.2. 安全解析手法 STPA

システムアクシデントの根本原因を、機器の故障や人間のオペレーションミスに置く、従来のアクシデントモデルでは、システムのアクシデントの可能性が潜在している状態（すなわちハザード）とその要因を事前に分析するための安全解析手法としては、FTA（Fault Tree Analysis）や FMEA（Failure Mode and Efect Analysis）の手法が用いられてきた。しかしながら、これらの手法は、現代の相互作用が複雑なシステムにおける安全解析手法としては十分ではなく、新しいアクシデントモデルによる分析手法が必要とされた。Leveson 教授が提唱した STPA（STAMP based Process Analysis）は、STAMP アクシデントモデルを前提として、システムのハザード要因を分析する新しい安全解析手法である。

STPA の手順は第 2 章に詳述するが、ここでは、手順の概略を述べる。STPA では、Step.0（前準備）でシステムが安全を実現する大まかな構造を分析した後、Step.1 でハザードに至るシナリオを、Step.2 でハザードの詳細要因を分析する。

Step 0：（準備 1）アクシデント、ハザード、安全制約の識別

対象システムにおいて分析対象となる、アクシデント、ハザード（アクシデントが潜在している具体的な状態）を定義し、ハザードを制御するためのシステム上の安全制約を識別する。

Step 0：（準備 2）コントロールストラクチャーの構築

システムにおいて、安全制約の実現に関係するコンポーネント（サブシステム、機器、組織等）、及び、コンポーネント間の相互作用（コントロールアクション、フィードバックデータ）を分析し、制御構造図（コントロールストラクチャー図：Control StructureDiagram という。以下、コントロールストラクチャー）を構築する。

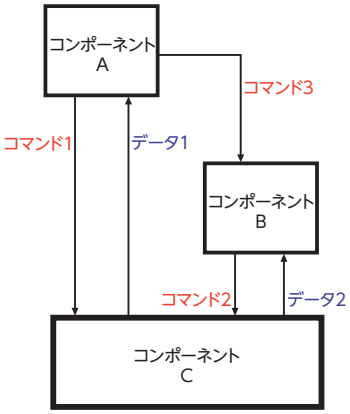


図 1.2-1　コントロールストラクチャーの例

Step 1：非安全なコントロールアクション（UCA）の抽出

コントロールストラクチャーから、安全制約の実行に必要なコントローラーによる指示すなわちコントロールアクションを識別し、4 種類のガイドワードを適用して、ハザードにつながる非安全なコントロールアクション（Unsafe Control Action：UCA という）を抽出する。

表 1.2-1　非安全なコントロールアクション（UCA）の抽出例

コントロールアクション	非安全なコントロールアクション			
	与えられないとハザード	与えられるとハザード	早すぎ、遅すぎ、誤順序でハザード	早すぎる停止、長すぎる適用でハザード
コマンド 1	XX 条件下で、コマンド 1 が提供されない場合、ハザードに至る (UCA1)	コマンド 1 の内容が誤っていた場合、処理は停止するが、ハザードには至らない	コマンド 1 の提供が、コマンド 2 よりも遅れた場合、指示が上書きされ、ハザードに至る (UCA2)	コマンド 1 が途中で停止した場合、ハザードには至らない
コマンド 2	コマンド 2 が提供されない場合、処理が開始しないが、ハザードには至らない	コマンド 2 の内容が誤っていた場合、処理は停止するが、ハザードには至らない	コマンド 2 の提供が、連続した場合、指示が累積され、ハザードに至る (UCA3)	コマンド 2 が途中で停止した場合、ハザードには至らない
コマンド 3	コマンド 3 が提供されない場合、処理が開始しないが、ハザードには至らない	コマンド 3 の内容が誤っていた場合、処理は停止するが、ハザードには至らない	コマンド 3 の提供が、遅れ／早い場合、処理開始がずれるが、ハザードには至らない	コマンド 3 が途中で停止した場合、ハザードには至らない

Step 2：ハザード要因（HCF）の特定

Step1 で抽出した非安全なコントロールアクション毎に、関係するコントローラーと被コントロールプロセスを識別して、コントロールループ図を作成し、ガイドワードを適用してハザード要因（Hazard Causal factor：HCF という）を特定する。特に、ソフトウェアや人間に起因する要因として、コントローラーの想定するプロセスモデルが、実際のプロセスの状態と矛盾することで起きる要因を特定する。

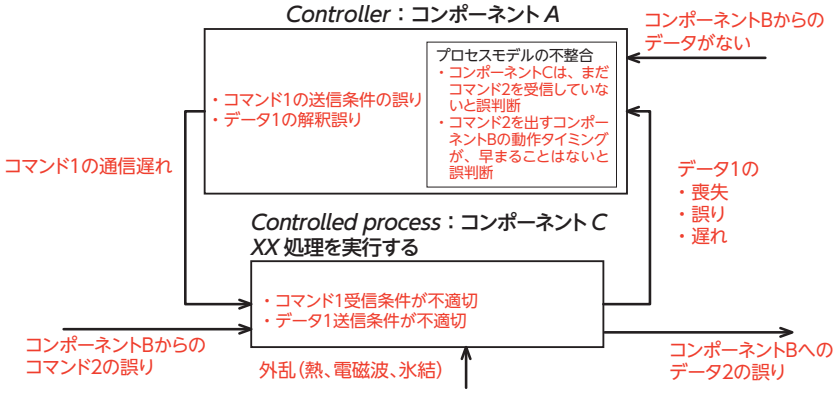


図 1.2-2　ハザード要因（HCF）の抽出例

STPA と従来手法との差異を以下に示す。

表 1.2-2　従来手法との比較

手法名	分析方法	特徴
従来手法 （FTA、FMEA）	●フォールトツリー図や影響分析表を用いてハザード要因を分析する ●システムの構成要素と故障モードが決まるアーキテクチャ設計の段階から適用できる	●機器や組織の単一故障をハザード要因として識別する ●分岐条件を論理的に組むことで網羅的に分析できる ●深く分析できる反面、全体的な視野での分析が難しい
STAMP/ STPA	●コントロールストラクチャーとコントロールループ図を用いてハザード要因を分析する ●システムの大まかな構成要素が決まる概念設計の段階から適用できる	●複数の機器や組織（人間）が、相互作用を行う複雑なシステムにおいて、相互作用のハザード要因を識別する ●過去のアクシデント事例データに基づくガイドワードにより網羅的に分析できる ●システム全体の振る舞いを確認しながら分析ができる

2. STPA の手順（全体説明）

2.1. 概要紹介

第 1 章での説明のように、STAMP では、安全に関するコントロールストラクチャーがシステムの安全制約を守れず、以下のような原因でハザードが発生することでアクシデントが発生すると考える。

【ハザード発生原因】

- 対処されない環境外乱や環境条件
- 対処されなかったりコントロールされなかったりするコンポーネントの障害
- コンポーネント間の非安全な相互作用
- 適切に協調されていない複数のコントローラーによるコントロールアクション

STAMP 自身はアクシデントを説明するモデルであり分析技法ではないけれども、STAMP をベースとして、解析の道具立てやプロセスがいくつか提案されている（図 2.1-1 参照）。

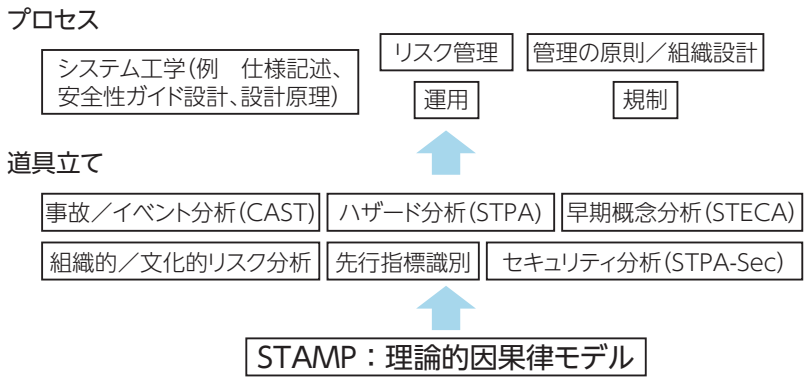


図 2.1-1　STAMP に基づく分析の道具立てとプロセス [STPA2015]

STPA はその一つでシステム開発を行う際などに用いるハザード分析法である [STPA2015]。第 1 章で述べられたような特徴を持つアクシデントモデル STAMP に基づく分析法であり、旧来のハザード手法とくらべソフトウェアの役割が大きなシステムの分析に適したものとされている。

STAMP/STPA では、まず対象システムを理解するといった準備の後、回避すべきアクシデントとハザードを決めて STAMP によるモデリングを行い STPA の分析に着手する。詳しくは提唱者である Leveson 教授らがチュートリアルとして “An STPA Primer”[STPA2015] を作成して公開しており、ここではその概要を述べる。STAMP/STPA では、STAMP モデルの以下の三つの要素を用い、コントロールを行う側とその対象プロセスとの間の相互作用において、安全制約が不適切であったり、守られない状態になったりするシナリオを中心に分析する。

【STAMP モデルの 3 要素】
- 構成要素間の相互作用を表すコントロールストラクチャー
- コントロールする側がその対象プロセスをコントロールするアルゴリズムと対象プロセスを（抽象化して）表現したプロセスモデル
- 守るべき安全制約

一般に、ハザード分析はあらゆるシステム安全プログラムの中心であり以下のような目的で行われる [Leveson2011]。

【ハザード分析の目的】
- 要件や設計の制約の開発
- 安全に対して要求や設計の妥当性の確認
- 運用の手順や指示書の準備
- テスト計画や評価
- 運営管理計画

また、分析が行われる段階という観点から、以下のようなタイプ分けがある。

【ハザード分析のタイプ】
- 予備的なハザード分析
- システムハザード分析
- サブシステムハザード分析
- 変更と運用の分析

STPA のプロセスは次のような 4 つのパートを含むとされている。

【STPA のプロセス構成】
- 分析やシステム開発に必要なシステムエンジニアリングの基盤を整える
- 可能性のある非安全なコントロールアクションを識別する
- 識別した非安全なコントロールアクションを使って、安全要求や安全制約を求める
- ハザードをもたらす各コントロールアクションがどのように生じるかを分析する

チュートリアルである "An STPA-Primer"[STPA2015] の中で著者らも述べているように、STPA は典型的にはトップダウンのアプローチで詳細化やシステムの発展に伴い繰り返し行われるものである。手順例として参照できるものはあるが、絶対的なものはなく、ハザード分析を行う目的や段階の違いも含め、STAMP/STPA の具体的な適用には様々なバリエーションがあり得る。ここでは IPA より公開されている日本語の報告書『STAMP 手法に関する調査報告書』[IPA2015] で紹介されている以下の手順を基本に説明する。

【STPA の手順例】
- Step0 準備 1：アクシデント、ハザード、安全制約の識別
- Step0 準備 2：コントロールストラクチャーの構築
- STPA Step1：安全でないコントロールアクション（Unsafe Control Action：UCA）の抽出
- STPA Step2：非安全なコントロールの原因の特定

2.2. Step0 準備 1：アクシデント、ハザード、安全制約の識別

最初のステップとして、アクシデント、ハザード、安全制約の 3 つを決める。これらを決めるのは他のシステムエンジニアリングのハザード分析にも共通するが、STAMP/STPA でのハザードの定義は、他の分析法のものと異なっている可能性がある点に注意する必要がある。

【アクシデント、ハザード、安全制約の定義】

- アクシデント

  望んでもいないし計画もしていない、損失につながるようなイベント。損失は、人命喪失、けが、物損、環境汚染、ミッション喪失、経済的損失といったものである。STAMP/STPA では安全工学の技法を出来るだけ広く適用する意図で、アクシデントも広めの定義としている。

- ハザード

  環境のある最悪な条件と重なることでアクシデントにつながるような、システムの状態もしくは条件。この定義には、安全性と信頼性の混同を回避するために単なる故障とハザードを区別し、分析可能性も高めるといった意図があり、以下のような二つの点を念頭においた定義となっている。まず、ハザードがコントロール対象のシステムの境界内のものであることを定義の後半で述べている。コントロール対象のシステムの範囲、コントロール範囲外の環境との境界が明確になっている必要がある。また、ハザードが実際に損失につながるのは、ハザードと組み合わさる、最悪の環境条件の存在が必要であることが定義の前半で述べられている。

- 安全制約

  ハザードが識別されると、それらからシステムを安全に保つための要件もしくは制約を導ける。トップダウンに考える場合、まず高レベルの安全制約が導かれることになる。安全制約はこの段階ですべて確定するとは限らず、STPA の実施によっても導出、修正されることもある。

2.3. Step0 準備 2：コントロールストラクチャーの構築

アクシデント、ハザード、安全制約を考えるのはアクシデントのモデルやハザードの解析法によらず一般的な手順である。しかしながら、コントロールストラクチャーを構築するのは STAMP/STPA のユニークな点となっている。システムのコンポーネントに機能を割り当てるといったことは一般のシステムエンジニアリングの活動でも行われるが、そのようなエンジニアリングのドキュメントとしてコントロールストラクチャーを使うことは一般に行われていない。

多くの複合的で複雑なシステムは、詳細な物理的設計の記述やドキュメントがあるにしても、機能的な振る舞いを把握するための情報はまとまっていなかったり、理解することが難ししかったりする。STAMP のコントロールストラクチャーは、システムの機能的な設計をうまくグラフィカルにモデルとして表現するもので、優れたドキュメントとして多くの人に役立つものである。

これまでの知見では、コントロールストラクチャーを作成したり、理解したりするには、まず簡潔な高レベルのものから始め、段階的に詳細化したり追加したりするのがよいとされている。最初の段階では、コントロール対象のプロセスとコントローラーだけ、場合によっては自動化や人がかかわるコントローラーの階層が若干加わった程度のコントロールストラクチャーから始めるのが典型的である。

2.4. Step1：非安全なコントロールアクションの抽出

典型的には STPA は以下の二つにステップを分けることが多い。

・非安全なコントロールアクションの抽出
・非安全なコントロールアクションの原因の特定

必ずしもこのステップの分割に従う必要はないが、ここではその分割に従った第一ステップの説明を行う。

コントロールストラクチャーをベースに、制御対象のプロセスに対するコントローラーからのアクションのうち、以下の 4 つのタイプの安全でないコントロールアクションを抽出する。

1. （**与えられないとハザード：Not Providing**）安全のために必要とされるコントロールアクションが与えられないことがハザードにつながる。
2. （**与えられるとハザード：Providing causes hazard**）ハザードにつながる非安全なコントロールアクションが与えられる。
3. （**早過ぎ、遅過ぎ、誤順序でハザード：Too early/too late, wrong order causes hazard**）安全のためのものであり得るコントロールアクションが、早すぎて、遅すぎて、または順序通りに与えられないことでハザードにつながる。
4. （**早過ぎる停止、長過ぎる適用でハザード：Stopping too soon/applying too long causes hazard**）（連続的、または非離散的なコントロールアクションにおいて）安全のためのコントロールアクションの停止が早すぎる、もしくは適用が長すぎることがハザードにつながる。

ハザードにつながる 5 番目のタイプとして、必要なコントロールアクションが与えられたけれども追従されない、という場合もあるが、この 5 番目の可能性については次のステップで分析する。

Step1 の分析に使うフォーマットに決まりはないが、非安全なコントロールアクション（Unsafe Control Action、以下 UCA と略記）のドキュメント化のためには、次のような表を用いるのが便利とされている。

表 2.4-1　Step1 UCA 識別に用いる表の例

コントロールアクション	与えられないとハザード	与えられるとハザード	早すぎ、遅すぎ、誤順序でハザード	早すぎる停止、長すぎる適用でハザード
（コントロールアクション）	（条件）	（条件）	（条件）	（条件）
…	…	…	…	…

UCA は、安全のために必要なルールと関連付けて考えることができるため、UCA の分析から安全制約を作り出すことも可能である。このため、安全制約を最初の準備の段階でなくここで作成したり、作成済みの安全制約を UCA の分析結果をもとに見直したりすることもできる。

2.5. Step2：非安全なコントロールの原因の特定

UCA を抽出したのち、非安全なコントロール（につながるシナリオ）の原因を特定する。必ずしもこれらのステップを完全に別のものとして逐次的に行う必要はなく、一部の UCA を抽出した段階で原因の分析を行う場合もありえる。この二番目のステップで、前述の 5 番目のタイプのシナリオである、安全のために必要なコントロールアクションの不適切な実行を考慮する。

このステップは、分析者の経験と考察を最も必要とするため、ステップ 1 よりも事前に設定できるガイドラインは少ない。しかしながら、可能性のあるハザードの原因を取り除いたり、緩和したりするなどして安全制約を保つために必要な情報を分析する重要なステップである。

基本的に、安全制約を保つためのコントロールループやその構成要素を確認し、以下のような点を分析する。

1. プロセスモデルが正しくなくなってしまう原因も含め、どのようにして非安全なコントロール、ひいてはハザードにつながるかを、分析する。
2. 必要なコントロールアクションが与えられたにもかかわらず、適切に実行されないのかの原因を分析する。この分析結果は、（FTA, HAZOP, FMEA といった）既存の分析の結果を含むことになるかもしれない。

図 2.5-1 はコントロールループに齟齬の原因となりうるものの例である。ここでは文献 [STPA2015]、報告書 [IPA2015] に沿った番号付けを行っている。

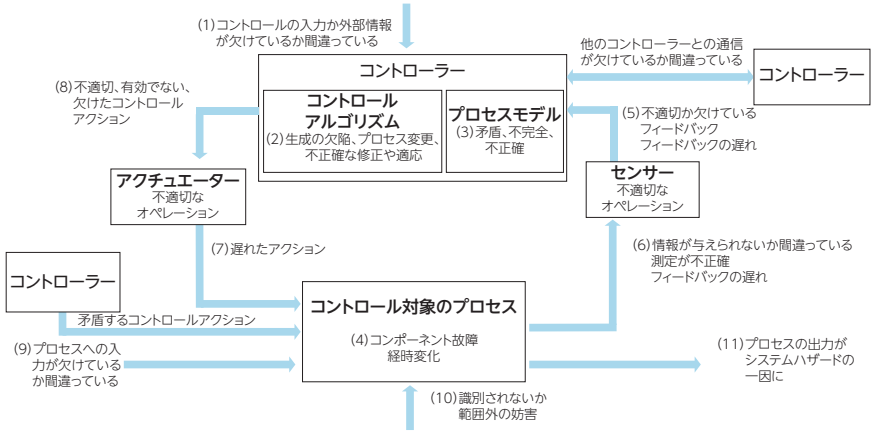


図 2.5-1　コントロールループで安全制約を破られる原因の例

（1）コントロール入力や外部情報の誤りや喪失
（2）不適切なコントロールアルゴリズム（作成時の欠陥、プロセスの変更、誤った修正や適用）
（3）不整合、不完全、または不正確なプロセスモデル。不適切な操作。
（4）コンポーネントの不具合。経年による変化。
（5）不適切なフィードバック、あるいはフィードバックの喪失。フィードバックの遅れ。
（6）不正確な情報の供給、または情報の欠如。測定の不正確性。フィードバックの遅れ。

（7）操作の遅れ。
（8）不適切または無効なコントロールアクション、コントロールアクションの喪失。
（9）コントロールアクションの衝突。プロセス入力の喪失または誤り。
（10）未確認、または範囲外の障害。
（11）システムにハザードを引き起こすプロセス出力。

分析によって得られたシナリオは、システムを保護するためのコントロールを識別するために用いられる。通常は、このような保護のためのコントロールの設計は STPA の解析そのものではない。STPA の解析結果を用いてドメインの専門家によってなされる。

注意点としては、単に図 2.5-1 のラベルだけを見て、一種の FMEA としてしまわないことである。本来の目標は、ループ内のコンポーネントの故障や不適切な操作を見つけることだけではなく、非安全なコントロールにつながり得るシナリオや問題の組み合わせを見つけることである。よくあるソフトウェアや人間の間違いの例は正しくないプロセスモデルによることが多いため、プロセスモデルがなぜ正しくなくなってしまうのかを考えることから始めるのが一つの方法である。

3. 対象システム概要

踏切は、警報灯・警報音鳴動装置および遮断機が設備されている一種踏切、警報灯・警報音のみで遮断機のない三種踏切、警報灯等もない四種踏切に分類される。なお、日本では、一種踏切（警報機・しゃ断機または係員を設置）が約 30,000 箇所、三種踏切（警報機のみ設置）が 800 箇所以上、四種踏切（警報機・しゃ断機なし）が 1300 箇所以上ある（平成 24 年 3 月末現在。なお、一部の時間のみ係員が操作する二種踏切は現在存在しない）。また、四種踏切は、踏切警標だけの踏切で列車の接近を知らせる装置は無く、制御システム要素は無い）。さらに、単線・複線、駅の中間にあるもの・駅の構内にあるもの等、様々な形態があり、それぞれが適切な制御を行っている。

今回、STPA を適用して安全解析を行う対象システムとしては「一種または三種踏切において、単線の駅中間踏切制御システム」を取り上げることとした。

踏切は列車の接近により鳴動を開始し、踏切道の通過により鳴動を停止するという非常にシンプルな仕組みである。また、あえて単線の踏切を選んだ理由は、列車の進入方向に応じた制御開始条件のマスク処理など、適度な複雑さを兼ね備えているということによる。

ちなみに、今回安全解析を行う WG 委員は鉄道の信号保安システムに関するドメイン知識を持っていない。また、事前に以下に示すような概要レベルの情報しか与えないことで「ドメイン知識を持たない技術者であっても STAMP/STPA の手法を活用することで安全解析を行うことができる」ことを実証してみることとした。

3.1. 単線の駅中間踏切制御装置の概要

図 3.1-1 に単線踏切の概念図を示す。実際の踏切は、列車の接近に伴い、警報灯が点滅し警報音が鳴動すると同時に遮断かんが降下し、通行者や車が入れないようにしている。今回は、列車接近の検知からの情報処理に着目し、踏切制御装置が遮断機等に対して警報開始命令を正しく指示できること、異常時でも安全に制御できることを検証した。すなわち、警報開始条件成立後の警報灯や警報音、遮断機の動作は分析の対象とはしていない。

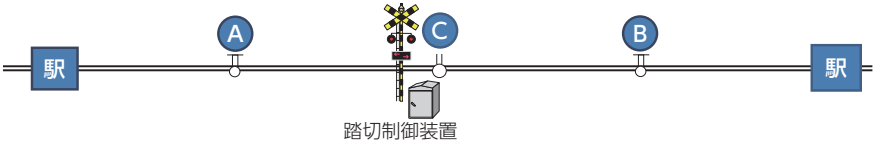


図 3.1-1　単線の駅中間踏切制御システム

図中の「A」および「B」は警報開始センサーであり、左右のレールに電気信号を流して電気回路として構成することで、列車の車軸によるレールの短絡により進来を検知する。「C」は警報終止センサーであり、同じく車軸によるレールの短絡を検知することで列車が踏切道を通過したことを検出する。なお、警報終止センサーは踏切の近傍に設置され、列車通過後一定時間経過後に警報終止とすることで、上り / 下りの条件を共用するものとする。なお、これらの列車検知センサーは、列車の走行方向の検出はできない。

3.2. 単線の駅中間踏切制御装置の要求仕様

単線の駅中間踏切制御システムに対する要求仕様（基本的な動き）を以下に示す。

ⅰ. 警報開始センサー（A,B）で列車を検知すると踏切鳴動を開始させる
ⅱ. 警報終止センサー（C）で列車を検知すると一定時間後に鳴動を終止させる
ⅲ. A から C に向かうとき、B をマスクする
ⅳ. B から C に向かうとき、A をマスクする

すなわち今回の分析対象は、以下のような動作が要求される制御システムである。

『図 3.1-1 にて列車が右向き（A から B の方向）に走行する場合、列車が A に差し掛かったら鳴動し、C を通り過ぎたら鳴動を停止する。その後、B を通過しても変化しない。同様に、列車が左向きに走行する場合、B に差し掛かったら鳴動、C を通り過ぎたら停止、A を通過しても変化しない。』

このような制御システムについて、ハザード要因分析を実施した。

4. STPA 分析実施例の説明

本章では、第 3 章に記した鉄道の実事例に対して本 WG が STAMP/STPA 分析を実施した結果を具体的に説明している。これまで、「STAMP とは」、「STPA とは」の解説書や、STPA による部分的な分析結果、あるいは最終結果抜粋のみの公開資料はあったが、STAMP 初心者向け解説書や STPA による分析手順を具体的に説明する日本語の公開資料は見当たらなかった。そこで、SEC では STAMP 初心者向けに、中間情報も含め、分析結果をすべて記載して説明することにした。この分析実施例は分析結果としての正解ではないことに注意願いたい。分析手順の参考として欲しい。

本分析の手順は SEC が公開した「STAMP 手法に関する調査報告書」[IPA2015] に記載の手順に則って進めた。

以下、4.1 を除いた各節は分析実施順番になっており、各節では、**【Input/Process/Output】、【実施例】、【勘所】**にまとめて説明している。

【Input/Process/Output】では、その分析作業を行なうのに必要な情報（Input）、行なうべき作業内容（Process）、作成して次の Step に渡すべき情報（Output）を整理した。

【実施例】では、各 Step で本 WG が実際に分析実施した結果の Output を記載した。

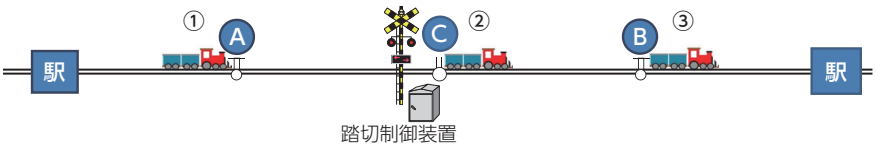
【勘所】では、各 Step を実施する際に気をつけるべきことを勘所としてまとめた。

4.1. 前提条件を整理する

前提条件は実施順番として最初に整理するのではなく、分析を進めながら、都度必要に応じて最小限の前提条件を策定（仮に前提を置く）していく。

UCA 識別、HCF 特定の Step において、与えられた要求仕様だけではコントロールアクションが Safe か Unsafe か、Hazard につながるか否かを判断できないことがある。判断基準がないまま分析を実施すれば、UCA の抽出において、どのコントロールアクションにどのガイドワードを当てはめても Safe にも成り得るし、Unsafe にも成り得るため、ほぼすべての組み合わせが UCA となってしまう。その後の HCF 特定においてもケースが異常に多くなり思考が発散してしまう。不足している判断基準の情報は、要求仕様の不足であるかもしれないし、実装依存かもしれない。いずれにしろ、何らかの仮定をしなければ分析できないため、こうなるべきであろうという仮定（分析における前提条件）を立てる必要があった。

本分析では、線路上を走行する列車の動きに応じて発生する事象と踏切制御システムが行なう制御を時系列に整理して、第 3 章で与えられた要求仕様から図 4.1-1 のように仮定した。



i. 警報開始センサー（A,B）で列車を検知すると踏切鳴動を開始させる
　→列車が開始センサー（A,B）に**到達**したら、鳴動開始させる
ii. 警報終止センサー（C）で列車を検知すると一定時間後に鳴動を終止させる
　→列車の最後尾が終始センサー（C）を**通過**したら、鳴動終始させる
iii. A から C に向かうとき、B をマスクする
　→列車の最後尾が終始センサーを**通過**したら、センサー B をマスクする
iv. B から C に向かうとき、A をマスクする
　→列車の最後尾が終始センサーを**通過**したら、センサー A をマスクする

図 4.1-1　実施例：要求仕様と仮定

i を「通過」にすると鳴動開始指示タイミングが列車の連結車両数や速度に依存してしまい、ii を「到達」にすると列車の連結車両数や速度によっては通過中に鳴動してしまうため、上記のように仮定した。

この仮定を、
・列車が線路走行中に発生する事象
・踏切制御装置への入力データ
・踏切制御装置が行なう制御（指示）
・被制を受けるコンポーネントの状態
に関するタイミングチャートで表し、図 4.1-2 のような機能仕様とした。

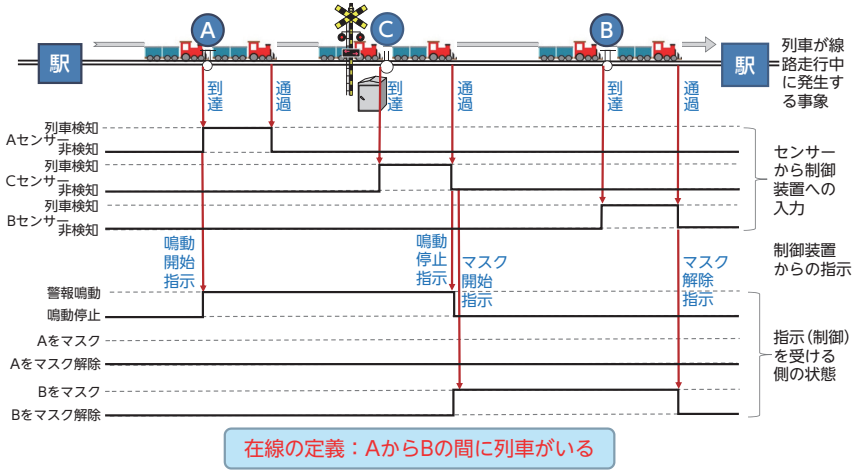


図 4.1-2　実施例：警報鳴動開始・停止指示の機能仕様

本 WG では、制御システム一般に関する経験・知識を有するが、鉄道ドメイン有識者ではない WG 委員がこの前提条件となる機能仕様を策定した。前提条件を策定する段階で設計を行なうのではなく、このようにしか成り得ない、という制御システム一般の常識の範囲に留めることが肝要と思われる。

4.2. Step0 準備 1　アクシデント、ハザード、安全制約の識別

ここで整理した安全制約が、次の Step1 UCA 識別への input となる。まず分析に先立って、アクシデントの定義もここで行う。分析の目的をこの Step で明確化する。

【Input/Process/Output】

アクシデント、ハザード、安全制約を識別するために必要な情報（Input）、行なうべき作業内容（Process）、作成して次の Step に渡すべき情報（Output）を整理する。

表 4.2-1　手順：Step0 準備 1　アクシデント、ハザード、安全制約の識別

作業名称	アクシデント、ハザード、安全制約の識別
目　　的	アクシデントを定義する 安全制約を導き出す
入　　力	①要求仕様書 ②［ドメイン専門家］
処　　理	①分析しようとするアクシデントが何であるかを定義する ②アクシデントと成り得るハザードには何があるかを考える ③ハザードの裏返しとなる程度の粒度で安全制約を導き出す
出　　力	①アクシデント、ハザード、安全制約の一覧表
備　　考	・アクシデント：喪失（Loss）を伴うシステムの事故 ・ハザード：アクシデントにつながるシステムの状態 ・安全制約：システムが安全に保たれるために必要なルール 例えば、踏切制御システムにおいて。 踏切がいつまでも開かないのは、サービス利用者･提供者に経済的損失を与えたり、精神的苦痛を与えることになることもあるが、人の生命に関わる事柄に焦点を絞ったときには『アクシデントではない』と定義できる。

【実施例】

この実施例では、交通渋滞をアクシデントから除外し、列車と人・車の衝突をアクシデントと定義したので、表 4.2-2 の赤枠内がアクシデントである。本 WG では人命に関わることを最重要課題と捉えたからであって、経済的損失を最重要視するなど、分析の目的によっては交通渋滞をアクシデントと定義することになるであろう。

表 4.2-2　実施例：Step0　アクシデント、ハザード、安全制約の識別

アクシデント（Loss）	ハザード（Hazard）	安全制約（Safety Constraints）
（A1）列車と人・車が踏切内で衝突する	（H1）列車が在線中に踏切が閉まらない（警報が鳴らない）	（SC1）列車が在線中は踏切が閉まらなければならない
（A1）列車と人・車が踏切内で衝突する	（H2）踏切遮断後、列車が在線中に踏切が開く（警報が鳴り止む）	（SC2）列車が在線中は踏切が開いてはならない
踏切が開かず、交通が渋滞する	列車が不在なのに踏切が閉まる（警報が鳴りだす）	列車が不在ならば踏切を閉じない
踏切が開かず、交通が渋滞する	列車が通過したのに踏切が開かない（警報が鳴り止まない）	列車が通過したら踏切を開ける

【勘所】

安全制約はハザードの裏返しのレベルの抽象度にする。この Step では抽象度が下がらないように注意して、要求仕様書に記載の抽象度に合わせる。

安全制約の抽象度が下がるとこの分析の段階で設計することになり、機能仕様を作成することになってしまいがちである。そうなると、具体的になる半面で漏れの発生が危惧される。安全分析においては想定外を排除するように気をつける。

4.3. Step0 準備 2　コントロールストラクチャーの構築

STAMP において非常に重要な制御構造図（Control Structure Diagram）を作成する。

Input となる要求仕様書から登場人物を整理して、制御の関係を明確化する。この Step では、フィードバックのデータの流れも制御と見做す。既存システムや類似システムに精通し、具体的な構成要素を熟知していると、つい詳細な制御構造図を書いてしまいがちだが、この Step においても抽象度を下げないことが重要である。

【Input/Process/Output】

コントロールストラクチャーを構築するために必要な情報（Input）、行なうべき作業内容（Process）、作成して次の Step に渡すべき情報（Output）を整理する。

表 4.3-1　手順：Step0 準備 2　コントロールストラクチャーの構築

作業名称	コントロールストラクチャーの構築
目　　的	登場人物間の依存関係を制御構造図で表す。 制御主体と制御対象の間で行われる制御（サブシステム間の相互作用）には何があるかを明確化する。 その後の分析作業において理解し易いイメージを共有する。
入　　力	①要求仕様書
処　　理	①要求仕様書から登場人物（ブロック）を抽出する ②要求仕様書から各ブロックの役割を抽出する。 ③役割を果たすために必要な制御、役割を果たした結果のフィードバックを抽出する。 ④制御、入出力情報（情報を与えるのみで制御を行うわけではない）の違いを分別する ⑤ブロック間を矢印線で結び、制御・入出力を・・(?) センサー出力のようなフィードバックを制御と考える ??
出　　力	①制御構造（コントロールストラクチャー）図
備　　考	ブロックの数は 4 つ程度が良いと言われている。 それ以上多くなる（抽象度を下げる）と、以降の分析すべき組み合わせが多くなり、集中しにくくなる、検討漏れを起こしかねないので、工夫が必要になる。

【実施例】

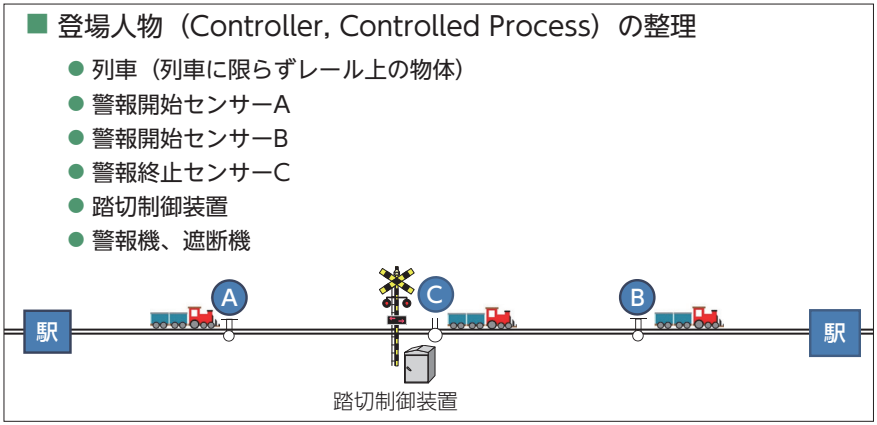


図 4.3-1　実施例：Step0　登場人物の整理

要求仕様に現れる登場人物を整理した。本分析では列車の運転士を登場人物に含めなかった。

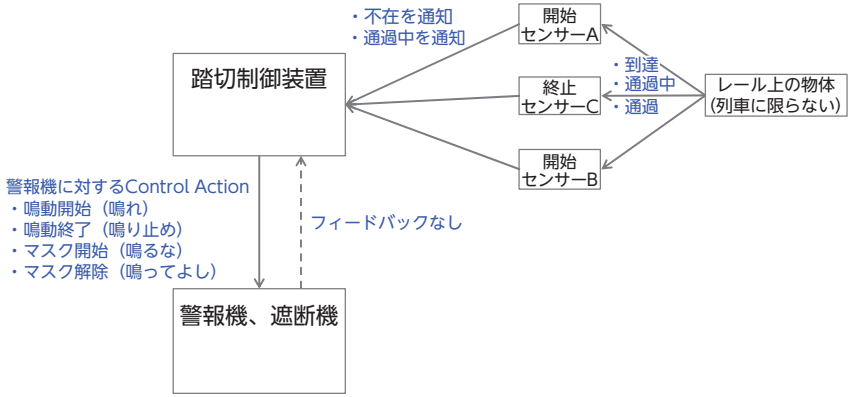


図 4.3-2　実施例：Step0　コントロールストラクチャーの構築

このコントロールストラクチャーでは、フィードバックがないのでコントロールループがない。

【勘所】

登場人物を整理する際には、要求仕様書に記載された登場人物のみをリストアップして、実装依存の登場人物が現れないようにする。

制御構造（Control Structure）に最終的な正解はなく、読む人に理解できる形になれば良い、と考える。なぜなら、STAMP とは説明のためのモデル（Explanatory Model）なのだから。

SysML やそのツールでは、事実を書き表す。一方、STAMP の制御構造図は注目する対象以外を削ぎ落として高い抽象度で記述する。よって、SysML で要求仕様を記述した結果がそのまま制御構造図として使えるとは限らない。但し、第 5.2 節で述べるように活用方法を工夫することによって作業効率改善や記述レベルの平準化（属人性を抑える）に役立つものと期待できる。

この Step の Output として制御の一覧が出来上がる。これが、Step1 の UCA 抽出の表（表 4.4-2）のコントロールアクションの列に入る。

各コンポーネント間の制御の関係を整理する際には、次のようにすると良い。

- 要求仕様から、各コンポーネントの役割を抽出する（コンポーネントの処理内容）
- 役割を遂行するために必要な制御・データを明確化する（コンポーネントへの入力）
- 役割を遂行した結果として、他コンポーネントや外部に出力する制御・データを明確化する（コンポーネントからの出力）
- 制御とデータは区別しやすいようする。制御は青、データは赤のように色分けしたり、制御は下向き矢印、データは上向き矢印に統一したり、など

コントロールストラクチャーにおける**ブロック（コンポーネント）の数は 4 つ程度**にするのが良いと言われている。

4.4. Step1　UCA（Unsafe Control Action）の抽出

【Input/Process/Output】

このStepでは、第2.4節の表2.4-1を用い、各制御（コントロールアクション）について、4つのガイドワードをヒントにして非安全なコントロールアクション（UCA）を抽出する。

表 4.4-1　手順：Step1　UCAの抽出

作業名称	UCA（Unsafe Control Action：非安全制御動作）の抽出
目　　的	ハザードにつながり得る制御動作の不具合を識別する（発想する）
入　　力	①UCAを導き出すための4つのガイドワード（4分類） ②アクシデント、ハザード、安全制約の一覧表 ③制御構造図
処　　理	①UCA識別の表を準備する ②最上列に4つのガイドワードを記す ③最左行に制御構造図中にある制御をすべて記す ④各マスごとに、当該（最左行の）制御動作が当該（最上列）状況になった場合、いずれかの安全制約違反に成り得るかを考える。 ⑤安全制約違反に成り得るならば、UCAであると判断する
出　　力	①縦軸：制御行動、横軸：ガイドワードとしたUCA一覧表。
備　　考	想定外を排除することを忘れないように。

【実施例】

コントロールストラクチャーを構築して得られた各制御に、STAMP特有の4つのガイドワードを当てはめて、前準備で識別した安全制約に違反するか否か識別した結果、表4.4-2のように6個のUCAが抽出された。

ここで、UCAではない場合、N/Aとはせず、どうして問題ないのかを記録しておくようにした。また、UCAについても次のStepで忘れないよう、メモ書きをした。

UCA3と5についてはガイドワードから直感的には発想しにくいが、図4.1-2**の状態・制御の組み合わせから考察する中で、あるいはStep2でシナリオを検討する中で発見できたものである。**

表 4.4-2　実施例：Step1　UCA の抽出

#	コントロールアクション	Not Providing	Providing causes hazard	Too early / Too late	Stop too soon / Applying too long
1	鳴動開始指示	(UCA1) 警報が鳴らずに列車が踏切を通過する(踏切が閉まらない)(SC1) 違反	列車が来ないのに警報が鳴る	(UCA2) 警報鳴動する前に列車が踏切に到達する(閉まるのが遅く間に合わない)(SC1) 違反	開始指示が継続するので、列車通過後に鳴動停止指示が出ても鳴動し続ける。
2	鳴動停止指示	列車が通過後も警報が鳴りっぱなし	(UCA3) 列車が通過中に鳴動停止する(SC2) 違反	(UCA3) 列車が通過完了する前に鳴動停止する(閉めた後、開くのが早すぎる)(SC2) 違反	(UCA1) 列車通過後も鳴動停止指示が続き、次の列車が来ても鳴動しない (開始指示と競合)(SC1) 違反
3	マスク開始指示	A、C を通過した列車が B に到達した時に再鳴動する	(UCA4) 列車が来ないのにマスク指示し、警報鳴動しない (UCA4) 反対側の開始センサーにマスク指示し、警報鳴動しない (SC1) 違反	(UCA5) 終始センサーへのマスク指示が遅れ、列車の当該センサー通過に間に合わないと、マスク指示が残り、対向列車が 2 本続いたときに警報鳴動しない (SC1) 違反	(UCA6) 列車が反対側の開始センサー通過後までマスク指示し続けると、対向列車が来ても鳴動しない (SC1) 違反
4	マスク解除指示	(UCA6) 反対側の開始センサーにマスク解除指示が出ず、対向列車が来ても鳴動しない(マスク指示後に列車が引き返す場合を含む)(SC1) 違反	警報が再鳴動する	列車が B を通過完了前に出ると再鳴動する	解除を後続列車によるマスク開始指示と競合するとマスクされずに再鳴動する可能性がある

【勘所】

- UCA がどの Hazard に至るか、あるいはどの安全制約に違反するかを明記すると良い
- Hazard に至らない場合も明記すると良い。空白や、N/A とはしない
- Safe/Unsafe の判断基準とした前提条件を明記する
- Step1 で作成する表は視認性が重要である。各欄にあまり多くを書かない。
- 記述の粒度は入手している情報のレベルとし、詳細にしない
- Context Analysis でも、支配的な変数に絞り、詳細は次の Step に送る

以前の STPA 解説書に記載されていた「Incorrectly Provided」は、「誤った制御情報が提供される」ではなく、「誤った条件で正しい制御情報が提供される」の意味なので、今は「Providing causes hazard」と記載するように改善されている。「誤った制御情報が提供される」は、主に故障やミスオペが要因であり、STPA の趣旨から外れるため積極的には扱わない、というのが STAMP 提唱者の考えである。

4.5. Step2　HCF（Hazard Causal Factor）の特定

【Input/Process/Output】

この Step では、Step1 で抽出した UCA について、どのような原因によってハザードと成り得るのか（安全制約違反になるか）を特定する。

表 4.5-1　手順：Step2　HCF の特定

作業名称	HCF（Hazard Causal factor：誘発要因）の特定
目　　的	どのような HCF があったら UCA に成り得るのかを考え、ハザードシナリオを作る
入　　力	① HCF 特定のための 11 個のガイドワード ②制御構造図 ③ UCA 一覧表
処　　理	①制御構造図からコントロールループを抜き出して、その中の各制御に該当するガイドワードを割り当てる ②［制御構造図中の各制御に該当するガイドワードを割り当てる］ ③ Step1 で識別した UCA 毎に、ガイドワードをひとつづつ当てはめてみて、ハザードと成り得るかを考える ④ハザードと成り得るならば、どういう条件下で当該ガイドワードの事象が発生して、その後、どういうシステム挙動になったらハザードとなって、アクシデントにつながるかのシナリオを作る
出　　力	①縦軸：UCA, 横軸：ガイドワードとした、ハザード要因の一覧表 ②ハザードシナリオ
備　　考	すべての UCA に夫々ガイドワードのすべてを当てはめて考える

第 2.5 節に記載した「コントロールループで安全制約を破られる原因の例」に、センサー、アクチュエーターの不適切動作を含めると、HCF を特定するためのガイドワードは次の 13 個になる。

（1）コントロール入力や外部情報の誤りや喪失
（2）不適切なコントロールアルゴリズム（作成時の欠陥、プロセスの変更、誤った修正や適用）
（3）不整合、不完全、または不正確なプロセスモデル。不適切な操作。
（4）コンポーネントの不具合。経年による変化。
（5）不適切なフィードバック、あるいはフィードバックの喪失。フィードバックの遅れ。
（6）不正確な情報の供給、または情報の欠如。測定の不正確性。フィードバックの遅れ。
（7）操作の遅れ
（8）不適切または無効なコントロールアクション、コントロールアクションの喪失。
（9）コントロールアクションの衝突。プロセス入力の喪失または誤り。
（10）未確認、または範囲外の障害
（11）システムにハザードを引き起こすプロセス出力
（12）アクチュエーターの動作が不十分
（13）センサーの動作が不十分

【実施例】

11 個のガイドワード（アクチュエーター、センサーの不十分な動作を含めると 13 個）をヒントに、非安全なコントロールアクションの原因を特定するが、フィードバックがないので適用するガイドワードは次の 7 つとした。

①上位からの指示や外部情報の誤り・欠落
②Control action が不適切・無効・欠落
③動作の遅れ
④プロセスへの入力の誤り・欠落

⑤意図しない、または範囲外の外乱
⑥不十分な制御・アルゴリズム
⑦部品故障・経年変化

これらのガイドワードをコントロールストラクチャーにマッピングすると下図のようになる。本来の手順ではコントロールストラクチャーからコントロールループを抜き出したもの夫々にマッピングするところだが、本事例にはコントロールループがないので、コントロールストラクチャーにマッピングした。

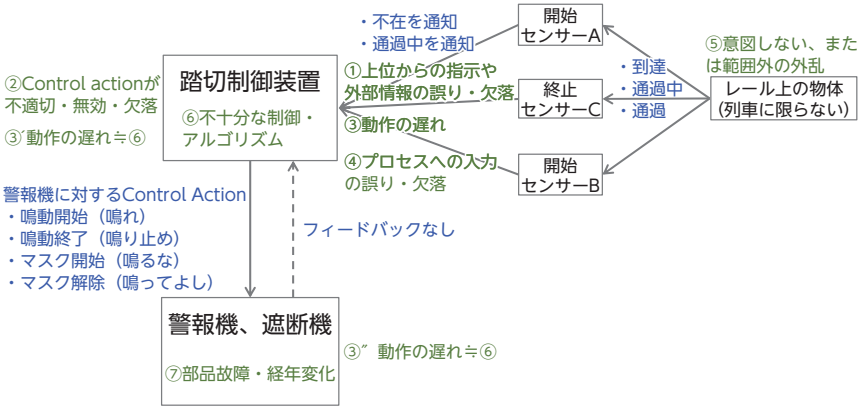


図 4.5-1　実施例：Step2　HCF を特定するガイドワードのマッピング

7 つのガイドワードを制御構造に対応させて誘発要因 HCF を特定する。尚、本分析ではコンポーネント間相互作用に注目したいため、警報機・遮断機は故障しないものと仮定し、部品故障を対象外としたので、⑦も除外した。

表 4.5-2　実施例：Step2　HCF の特定

	①上位からの指示や外部情報の誤り・欠落	②Ｃｏｎｔｒｏｌ action が不適切・無効・欠落	③動作の遅れ	④プロセスへの入力の誤り・欠落	⑤意図しない、または範囲外の外乱	⑥不十分な制御・アルゴリズム
(UCA1) 警報が鳴らずに列車が踏切を通過 (踏切が閉まらず)		・踏切通過後に引き返す列車向け制御が不適切 ・鳴動停止継続により次の鳴動指示と競合		センサー A が故障して A から踏切制御装置への通知が欠落		
(UCA2) 鳴動前に列車が踏切に到達 (閉まるのが遅い)			・警報機の動作遅れ			・踏切制御装置の動作遅れ
(UCA3) 列車が踏切を通過する前に鳴動停止 (開くのが早い)					列車が A を通過後、踏切に到達する前に、C が外乱により短絡する	
(UCA4) 不正なマスク開始指示が出て、列車が来ても警報鳴動しない		・踏切制御装置の状態管理が不適切				・踏切制御装置の状態管理が不適切
(UCA5) マスク解除指示漏れで、列車がきても鳴動せず			・超高速列車に対応できずにマスク解除の指示遅れ	・レール上の物体による外乱		・制御装置の処理に問題があり、マスク解除の指示遅れ
(UCA6) マスク開始指示漏れ	・誤った外部入力（外乱）でマスク解除漏れ	・制御装置の処理遅れでマスク解除漏れ	・状態制御誤りでマスク解除漏れ	・不正な外部入力によりマスク解除漏れ		・非定常運行への対策漏れでマスク解除漏れ

UCA の夫々について、前ページのガイドワードをすべて当てはめてみて、何があったら UCA に成り得るかを考える。「何があったら」のヒントがガイドワードである。

すべてのガイドワードに対して UCA に成り得るケースが思いつくとは限らない。逆に、ひとつのガイドワードから複数のケースが思いつくこともある。

STAMP 提唱者は、HCF 特定の際にチェックリストを使わないほうが良い、との意見である。理由は、自由な発想の範囲を制限しないため、である。確かに、発想する際にはチェックリストで縛らない方が良いかもしれない。しかし、業務で分析を実施したら、これだけ実施したというエビデンスを示す必要がある。網羅性を示すエビデンスとして、後追いでチェックリストに記入することは意味があり、業務遂行上は必要なことと思われる。本分析においても、表 4.5-2 は以下のシナリオ作成の後で作成した。

第 4.5.1 項から第 4.5.6 項では、ガイドワードから容易に発想できるシナリオには 🙂 を付記し、発想し難いシナリオには ✸ を付記した。更に、複雑なシナリオには事象・制御・状態のタイミングを表す図を記載した。

また、各シナリオに特化した解決策の案を記述した。

【勘所】

HCF を特定するためのガイドワードは汎用的な表現の文言である。実際に特定システムに関して HCF の特定をする際には、当該ドメインの用語に読み替えた具体例をいくつか例示しておくと良いと感じた。但し、その場合でも想定外を排除するため、元の汎用的な文言は残すべきであろう。

4.5.1.【実施例】UCA1 に至るハザードシナリオ

■（UCA1）警報が鳴らずに列車が踏切を通過する（踏切が閉まらない）安全制約 1 に違反

シナリオ 1-1「④プロセスへの入力の誤り・欠落」センサー A が故障して A から踏切制御装置への通知が全く届かない
- □ 対策：開始センサーからの信号が途絶えたら警報を鳴らす

難 シナリオ 1-2「④プロセスへの入力の誤り・欠落」センサー A が不電導物（葉っぱなど）に覆われて、車輪経由の検知電流が流れず、列車到達を検知できない
- □ 対策：一瞬でも短絡したら、短絡時間異常と判断し警報鳴動し続ける
- □ 対策：センサー検出順番の不正を検出したら警報鳴動し続ける

難 シナリオ 1-3「② control action が不十分」A から来た列車が C を通過した後、連結を切り離して、後部車両が A 方向に引き返す。
- □ 対策：通常運行において列車は退行してはならない

難 シナリオ 1-4「② control action が不十分」A から来た列車が C を通過した後、A 方向に引き返す。
- □ 対策：通常運行において列車は退行してはならない

難 シナリオ 1-5「② control action が不十分」A から来た列車が C を通過して B をマスクした後、B と C の中間で停止。救援列車が反対方向から侵入してセンサー B を通過。A 方向に進行する。

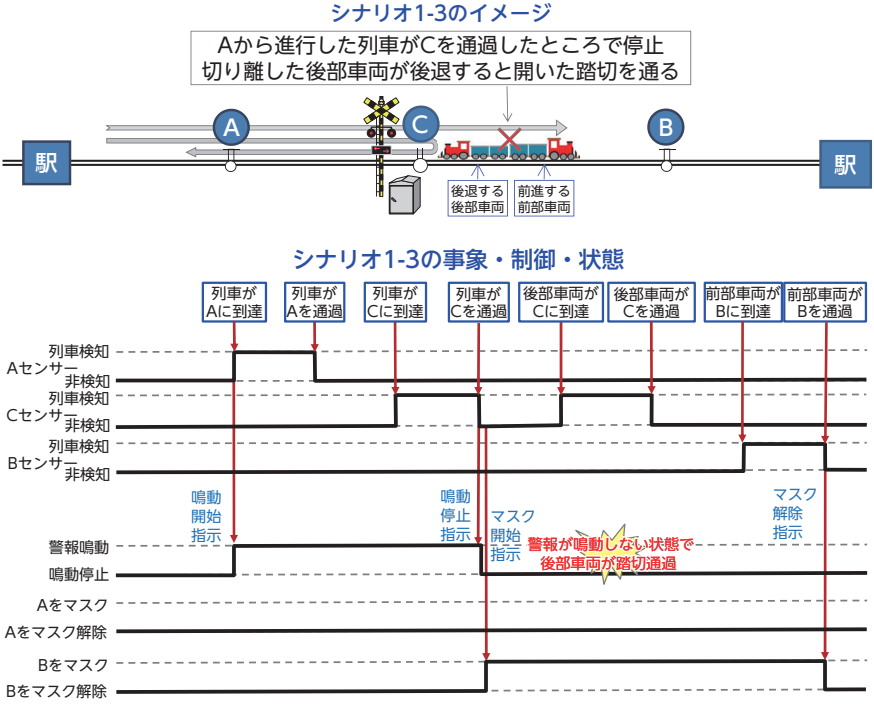


図 4.5-2　実施例：Step2 ハザードシナリオ作成　UCA1

HCF を特定する際、システムがどういう状態であるかを分かり易くするため、絵や図を用いると良い。この事例では、コンポーネント間の論理的関係を絵で表し、順序関係を事象と制御と状態に着目して時系列で表現してみた。

4.5.2.【実施例】UCA2 に至るハザードシナリオ

■（UCA2）警報鳴動開始する前に列車が踏切に到達する（閉まるのが遅く、間に合わない）安全制約 1 に違反

- シナリオ 2-1「③´動作の遅れ」列車が超高速なため、センサー A に到達後、警報機・遮断機が動作開始する前に列車が踏切に到達する
- シナリオ 2-2「③″動作の遅れ」警報機・遮断機動作開始が遅いため、センサー A に到達後、警報機・遮断機に警報鳴動開始指示がでる前に列車が踏切に到達する
- シナリオ 2-3「③´動作の遅れ」踏切制御装置のタイマーが遅延して、センサー A に到達後、警報機・遮断機に警報鳴動開始指示がでる前に列車が踏切に到達する

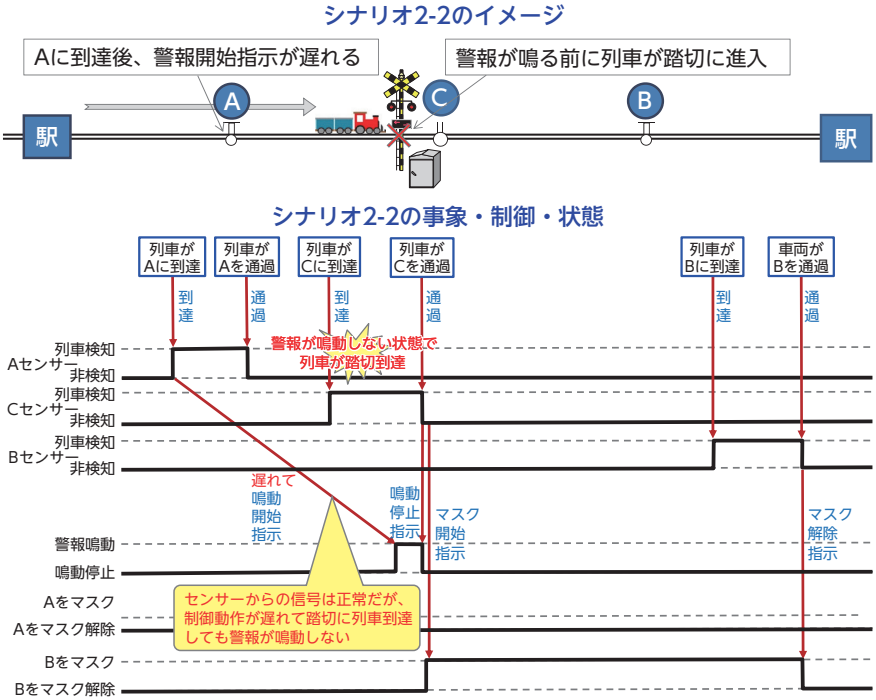


図 4.5-3　実施例：Step2 ハザードシナリオ作成　UCA2

警報開始センサー A に列車が到達したことを検知したが、何らかの理由により踏切制御装置からの警報鳴動開始指示が遅れて、警報が鳴動する前に列車が踏み切りに到達し、

安全制約 1 に違反する、というシナリオである。「何らかの理由」には何が有り得るのかは次の Step で検討する。この Step では想定外を排除しないよう、「何らかの理由」が有り得るか否かを考えないこととした。

4.5.3.【実施例】UCA3 に至るハザードシナリオ

■（UCA3）列車が踏切を通過完了する前に鳴動停止する（閉めた後、開くのが早すぎる）安全制約 2 に違反

難 シナリオ 3-1「⑤外乱」列車が A を通過後、踏切に到達する前に、C が外乱により短絡する（突風で金属製看板が飛び、C に被さり短絡し、その後 C から飛び去る）

□ 対策：列車の通過では有り得ないような非常に短時間の短絡を列車通過とは判断しない。異常な物体がレール上に存在するものと判断し、鳴動停止しない。

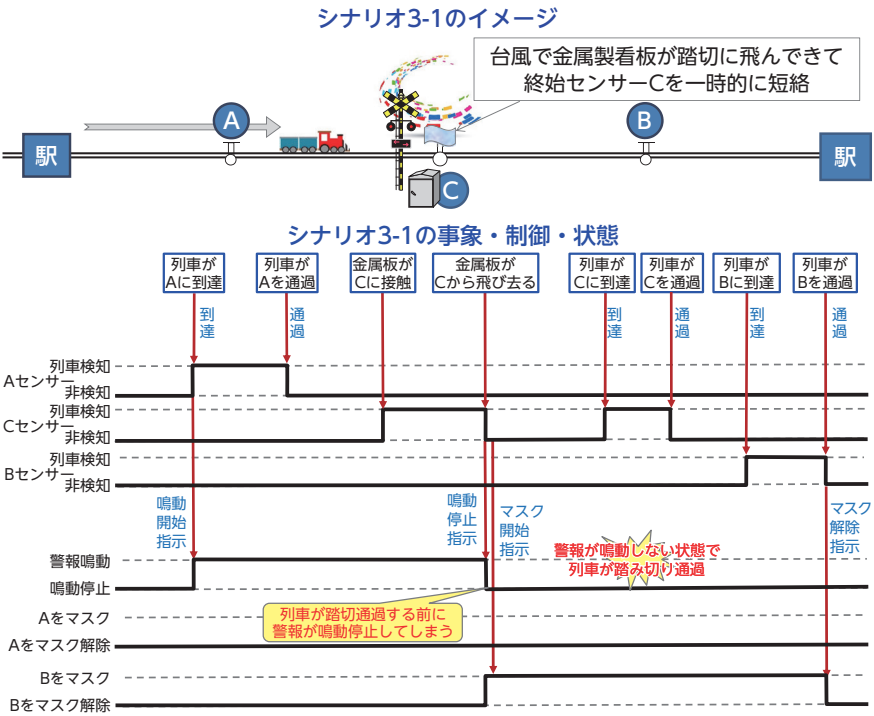


図 4.5-4　実施例：Step2 ハザードシナリオ作成　UCA3

このシナリオには、要求仕様に記載の無い登場人物 " レール上の物体 " として、風に煽られて飛ばされてきた金属性看板が登場する。このシナリオはガイドワードをヒントに容易に発想したものではなく、Step2 を実施している中でこういうタイミングになる事象も有るのでは、と発想したものである。

4.5.4.【実施例】UCA4 に至るハザードシナリオ

■（UCA4）列車が来ないのにマスク指示し、警報鳴動しない 安全制約 1 に違反
■（UCA4）反対側の開始センサーにもマスク指示し、警報鳴動しない 安全制約 1 に違反

シナリオ 4-1「②コントロールアクションが不適切 and/or ⑥不十分な制御・アルゴリズム」列車が A,C を通過して B のみをマスクすべきところ、A もマスクしたため、後続列車がきても警報鳴動しない。

□ 対策：マスク解除するときには、A と C の両方のマスクを解除する。踏切制御装置が状態を持つと、状態制御に関連する誤りが発生しうる。誤りがあっても対応できるようにする必要がある。

難 シナリオ 4-2「③動作の遅れ」+[⑥不十分な制御・アルゴリズム] A 方向から来たセンサー BC 間よりも長い列車が終始センサー C を通過中、開始センサー B が列車を検知し警報（再）鳴動する。終始センサー C を通過後 B へマスク指示を出し列車が B を通過後マスクを解除する。一方終始センサー C はセンサー B から検知を受けているため列車が通過後 A にもマスク指示を出すが、列車は A を通過することはないので、マスクが残り A 方向から後続列車が来ても警報鳴動しない。

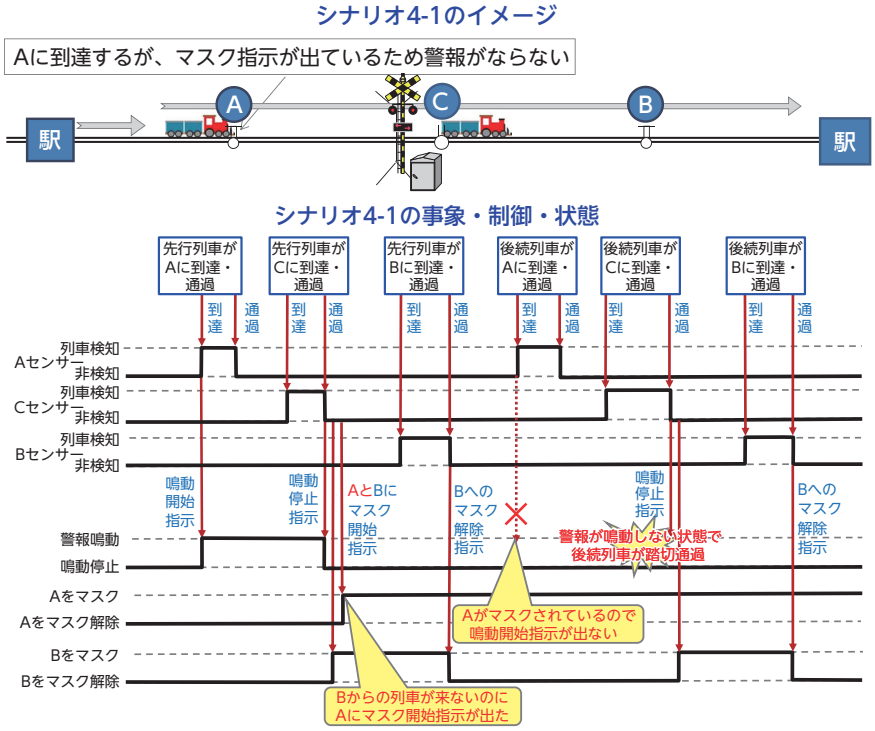


図 4.5-5　実施例：Step2 ハザードシナリオ作成　UCA4

このシナリオでは、マスク開始指示というコントロールアクションに関してガイドワー

ドを当てはめ、制御装置にガイドワード②または⑥の問題が有ったら両センサーへマスク開始指示をしてしまう可能性もあるのでは、と考えた。踏切制御装置内に具体的にどういう問題が内在していたらこのようになるかまでは考えないようにした。もし、それを考えるといろいろなケースが考えられる一方、考え漏れ（想定外）が発生することに成り得る。

4.5.5.【実施例】UCA5 に至るハザードシナリオ

■（UCA5）終始センサーへのマスク指示が遅れ、列車の当該センサー通過に間に合わないと、マスク指示が残り、対向列車が 2 本続いたときに警報鳴動しない 安全制約 1 に違反

雑 シナリオ 5-1「③動作の遅れ or ⑥不十分な制御・アルゴリズム」A 方向から来た列車が終始センサー C を通過後、開始センサー B へのマスク指示が遅れ、マスクする前に列車が B を通過すると、警報が再鳴動し、かつ B へのマスク指示が残る。その後、対向列車が 2 本続いたとき、2 本目の列車がきても警報鳴動しない。動作遅れ要因には、制御装置の動作遅延、列車の連結が非常に長い（C 通過による停止指示と B 通過による鳴動指示の競合）、または列車が超高速、が有り得る

□ 対策：長時間の短絡は異常と判断し、マスク解除し、警報鳴動する

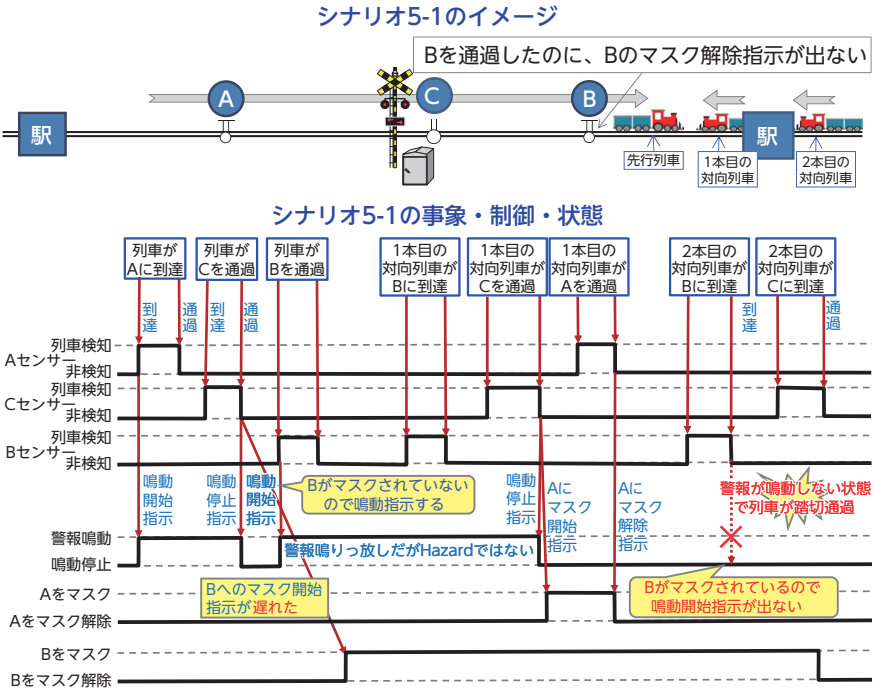


図 4.5-6　実施例：Step2 ハザードシナリオ作成　UCA5

このシナリオは、何らかの理由（ガイドワードの③または⑥）によって、開始センサー B へのマスク開始指示が遅れたケースである。

4.5.6.【実施例】UCA6 に至るハザードシナリオ

■（UCA6）列車が反対側の開始センサー通過後までマスク指示し続けると、対向列車が来ても鳴動しない 安全制約 1 に違反

難 シナリオ 6-1「③動作の遅れ」マスク解除指示が大幅に遅れる…シナリオ 5-1 と同じ

シナリオ 6-2「②コントロールアクションが欠落」マスク解除指示が出ない

シナリオ 6-3「⑥不十分な制御」マスク解除指示が届くが処理されない

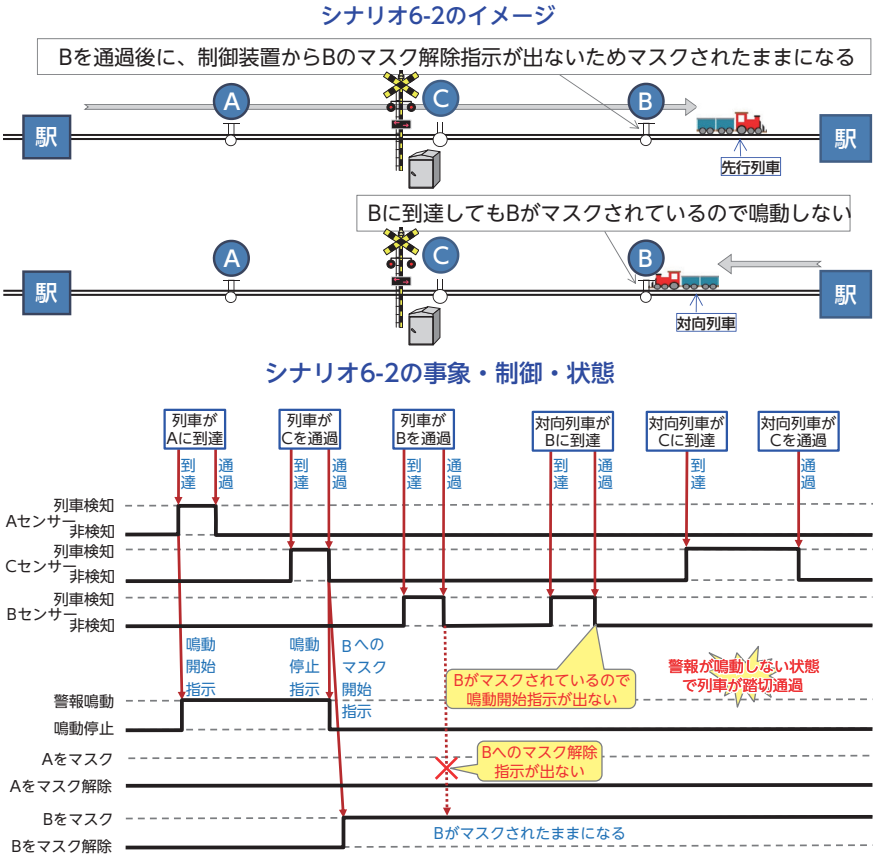


図 4.5-7　実施例：Step2 ハザードシナリオ作成　UCA6-2

ハザードシナリオとして記載していないが、上記シナリオの他に、A 方向から列車が来て B へのマスク指示後に列車が A 方向へ引き返すケースもある。その場合、B のマスク状態が残り、B 方向から対向列車が来たときに警報が鳴動しないという UCA は、「マスク開始指示が与えられるとハザードを引き起こす（Providing causes hazard）」UCA4 のようにも思える。しかし、本分析において「C を通過したら B をマスクする」と仮定

したので、A から来た先行列車が B に行かないなら B のマスクを解除するのが正しいコントロールアクションである。

よって、B へのマスク解除指示が行われない（Not providing causes hazard）という UCA であると考えて UCA6 に分類できる。

■（UCA6）**反対側の開始センサーにマスク解除指示が出ず、対向列車が来ても鳴動しない（マスク指示後に列車が引き返す場合を含む）安全制約 1 に違反**

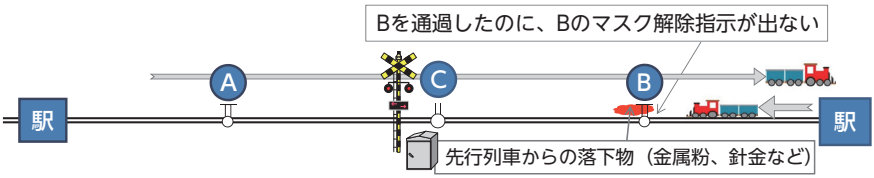
シナリオ 6-4「②コントロールアクションが不適切＋ ③動作の遅れ」制御装置からのマスク開始指示が遅れ（B を通過後に開始指示）再鳴動し、（かつ B のマスク状態が残った状態となり）対向から列車がきても鳴動しない

□ 対策：マスク解除は全開始センサーに対して行う

難 シナリオ 6-5「④プロセスへの入力の欠落」B を通過中に列車から針金、金属のチェーン、あるいは金属の粉などの電気伝導体が落ちて短絡し、いつまでも列車通過中と判断し、B のマスクが解除されない

□ 対策：長時間の短絡は異常と判断し、マスク解除し、警報鳴動する

シナリオ6-5のイメージ



シナリオ6-5の事象・制御・状態

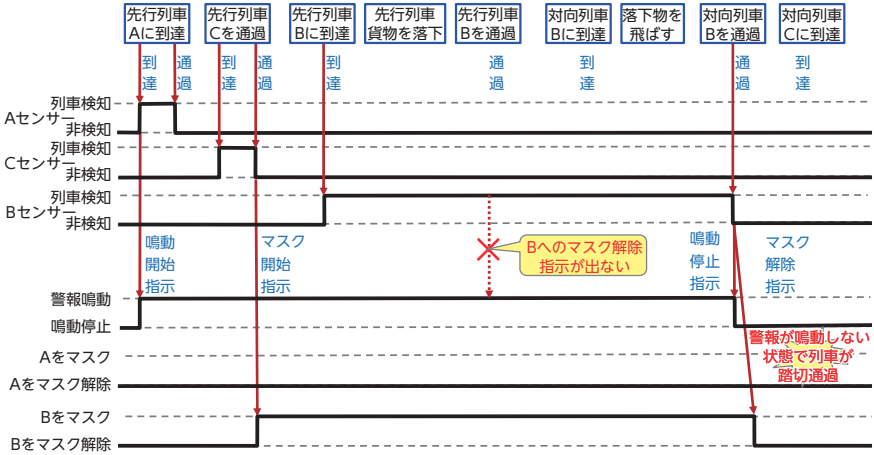


図 4.5-8　実施例：Step2 ハザードシナリオ作成　UCA6-5

このシナリオは図 4.5-8 のタイミングを先に考え、後からシナリオを作成した。

【勘所】

- 複数のコンポーネント間の相互作用を同時に考えるのではなく、二つのコンポーネントに絞って、各コントロールループに関して考えると良い。
- シンプルに考える。例えば、踏切制御では例えば、列車を除外する、など
- コントローラーの中に書いた誤りがシステム要員で重要（コントローラー内部の仕様の誤りを見つける）
- プロセスモデルの不整合の分析では、仕様入手すれば内部の状態が分かる

HCF の特定とは、HCF ガイドワードをヒントにして、原因と成り得るシナリオを発想する作業である。如何に漏れなく発想できるかが重要となるが、どうやって発想するかについては特にコツはない。もちろん、自由な発想ができる特殊な能力を持っていることが好ましいに決まっているが、その能力は先天的と思われる。特殊な能力が無いのならば、慣れることが一番と感じた。実際、当 WG での分析でも、UCA1 から UCA6 へ順番に分析して、後になればなるほど発想の仕方に慣れてきて、短時間に発想できた。

Step2 で新たな UCA を発見することもある。また、ある UCA についてシナリオ考察する中で別の UCA に関するシナリオを発見することもある。したがって、シナリオの考察から Step1 に戻って新たな UCA を追加することも重要である。

4.6. STPA 最終 Step：対策のまとめ

各シナリオに特化した対策はシステム全体としては矛盾するものもあり得るので、第 4.5 節の各シナリオに特化した対策を、システム全体として整合させて整理した結果が表 4.6-1 である。

この作業 Step は STAMP/STPA の手順として定義しているものではない。どのような分析手法を用いた場合でも、これをやらなければ分析が無駄になるので絶対に実施しなければならない作業 Step である。

表 4.6-1　実施例：対策のまとめ

#	対策	関連 UCA	関連 HCF	対策対象コンポーネント	備考
1	マスク解除は両方向の開始センサーに行なう	UCA4	4-1	踏切制御装置、開始センサー	
		UCA6	6-4		
2	終始センサーが列車 “ 到達 ” を検知したら、開始センサーへマスク指示する	UCA4	4-2	踏切制御装置、開始センサー	列車 “ 到達 ” を前提として再度 STPA 分析要
3	開始センサーからの信号が途絶えたら警報を鳴らす	UCA1	1-1	踏切制御装置、開始センサー	Heartbeat、Healthy 信号等による監視機能が必要
4	センサー検出順番の不正を検出したら警報鳴動し続ける	UCA1	1-2	踏切制御装置	順番の正誤判断基準要
5	異常に短時間の短絡は異常と判断し、警報鳴動し続ける	UCA1	1-2	踏切制御装置、開始 / 終始センサー	異常時間の判断基準要
		UCA3	3-1		
6	異常に長時間の短絡は異常と判断し、マスク解除し、警報鳴動する	UCA5	5-1	踏切制御装置、開始 / 終始センサー	異常時間の判断基準要、開始と停止の指示競合時の処理判断基準要
		UCA6	6-5	踏切制御装置、	
7	退行時には非常手続きが必要	UCA1	1-3 1-4	列車、運転士	外部コンポーネントも絡む

5. Advanced Technic

5.1. ツール活用

前述の報告書 [IPA2015] に紹介されているように、STAMP のハザード分析 STPA を支援するツールは既にいくつかある。ここではシュトゥットガルト大学ソフトウェア技術研究所（Institute for Software Technology, University of Stuttgart）の Asim Abdulkhaleq 氏が開発し公開している、XSTAMPP（An eXtensible STAMP Platform）を紹介する。このツールは当初 A-STPA（Automated tool support for STPA）と呼ばれていたものを発展拡張させたものである。

XSTAMPP は、Eclipse の Plug-in-Development Environment（PDE），Rich Client Platform（RCP）に基づいて Java でプログラムされている。XSTAMPP は STAMP モデルに基づく安全工学のベースプラットフォームとしての活用と発展を念頭に開発されており、STAMP モデルに基づく新しいアプローチや機能で容易に拡張可能なように作られている。本書の執筆時点（2016 年 2 月）で XSTAMPP のバージョンは 2.0 で次の三つのプラグインを含んでいる。

・A-STPA（Automated tool support for STPA）
・A-CAST（Automated tool support for CAST）
・XSTPA（Extended Approach to STPA）

ハザード分析 STPA や自己分析 CAST の基本的な支援する機能を持っているので、アクシデントとハザードの関連づけ、コントロールストラクチャーとコントロールアクションの関連付け、コントロールアクション分析表のテンプレート生成、コントロールストラクチャーへのプロセスモデルの追加、といった複数ステップにまたがる分析作業を統一的に行うことができる。また、ある作業ステップでの成果物に対する更新が、関連する情報法共有する他の作業ステップの成果物に反映される。そのためエクセルとパワーポイントのファイルを別々に開いて、一貫性を気にしながら作業するといったことを行う必要がない。また、XSTPA によりコンテクストテーブルの内容から LTL（Linear Temporal Logic）形式の仕様記述への変換や、組み合わせテストの支援といった機能を持つように拡張されている。

ここでは第 3.1 節で概要が述べられた単純な踏切の例題に対するハザード分析 STPA に焦点をしぼり、XSTAMPP の利用法の概略について説明する。進行制御システムが別途存在し、正常な駅間制御により対向列車の問題はないとしている。ソフトウェアで踏切を制御することを想定し、モデル化と分析は、以下のような方針で行う。

・通常、STAMP/STPA は抽象度の高いモデルからトップダウンに行う。最初のモデルとして制御対象プロセスが一つの単純なモデリングを行うことから着手する。
・コントロールアクションとして、警報装置への開始や停止だけでなく、センサーに対してマスク指示を発行するような、複数の制御対象があるモデルにしていない。センサーからの信号に対し、コントロールプロセス側のプロセスモデル内にマスク用のフラグを持ち、コントロールアルゴリズムでフラグを参照・管理するモデルとした。
・警報開始と警報停止は離散的なコントロールアクションとし、それらのコントロールアクションに対しては、非離散的なコントロールアクションを想定した、短すぎる適用、長すぎる適用といったハザードは考慮していない。

新しいプロジェクトの作成

XSTAMPP は Eclipse プラグインツールであり、ワークスペースを決めてプロジェク

トを作成して使用する。使い始めの場合で図 5.1-1 のような起動画面が表示されている場合、"Create new..." をクリックして新しいプロジェクトを作成開始する。もしくは "File" メニューの "Create New Project" を実行する。

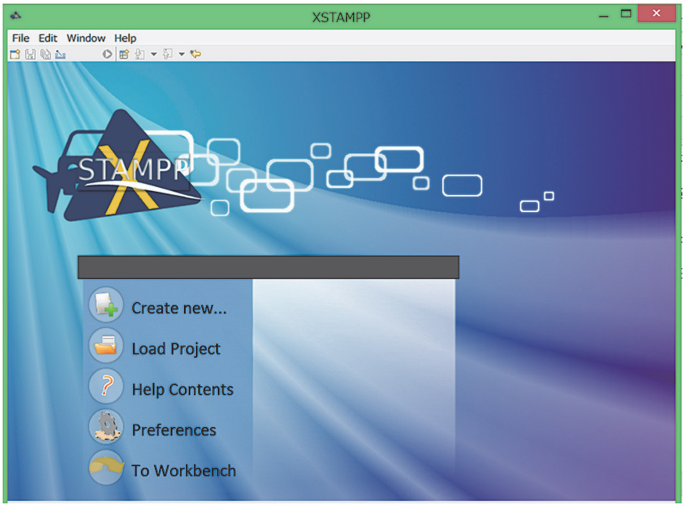


図 5.1-1　起動画面

図 5.1-2 のようなウィザードが立ち上がる。ここでは STPA 分析のため "STPA Projects" の中の "STPA Project" を選択して "Next" をクリックする。

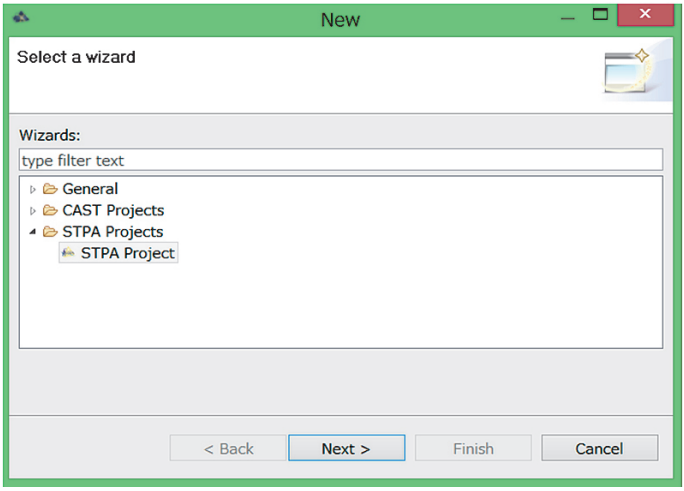


図 5.1-2　プロジェクトタイプの選択

次に図 5.1-3 のようなプロジェクトの名前やデータ形式を指定する画面になる。このとき、プロジェクトファイルの拡張子（データの保存形式）を選ぶ必要があるが、旧

バージョンとの互換性を保つために古い形式にする必要がなければ、新形式に対応する "hazx" を選び "Finish" をクリックする。

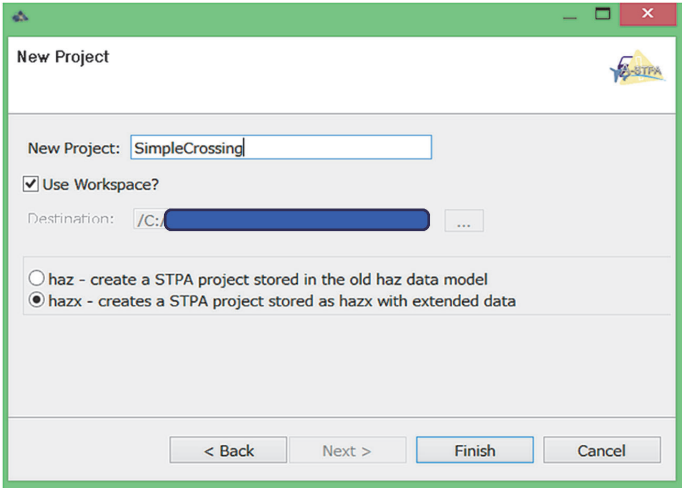


図 5.1-3　プロジェクト名とデータ形式の指定

プロジェクトが作成できると、その名前が Project Explorer にリストアップされる（図 5.1-4 参照）。プロジェクト名の左側の小さな三角をクリックすると、STPA 分析の三つの分析ステップ、"0 Establish Fundamentals", "1 Unsafe Control Actions", "2 Causal Analysis" が表示され、さらに各ステップの左側の小さな三角をクリックすると、図 5.1-4 のようにより具体的な詳細ステップが展開された表示となる。以下、それらのうちの主なものを、前述の踏切の例をベースに説明する。

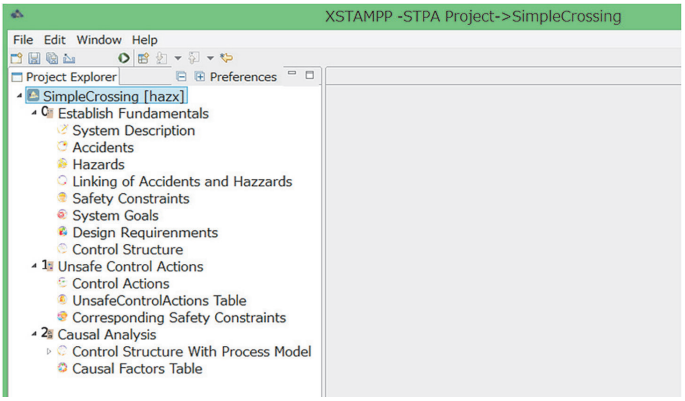


図 5.1-4　プロジェクトと分析ステップ

"0 Establish Fundamentals"

"Project Explorer" で "System Description" をクリックすると、システムの記述を記

入するエリアが開く。図 5.1-5 はそのエリアに第 3.2 節の記述を入力したものである。

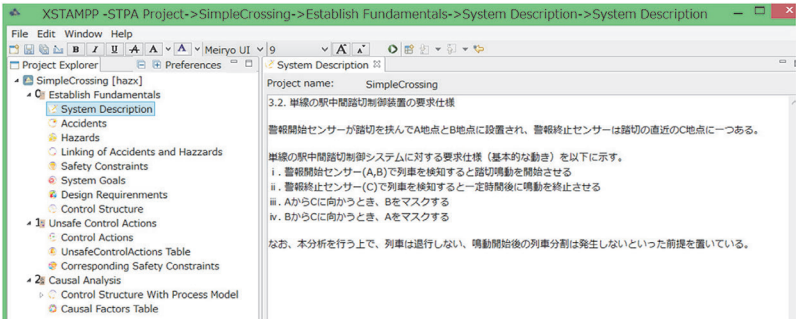


図 5.1-5　System Description ステップ

"Accidents" は、アクシデントとその説明を記述する（図 5.1-6 参照）。アクシデントの項目を追加したい場合は、 + をクリックすると識別番号が自動的に割り付けられ、アクシデントの "Title", "Description/Note" が記入できるようになる。図 5.1-6 はアクシデントとして、「列車と人・車が踏切内で衝突する」を記入したものである。"Links" は対応するハザードを記入する欄で、その説明は "Linking of Accidents and Hazards" で後述する。"Title" を編集したい場合は項目上でダブルクリック、"Description/Note" を編集したい場合はクリックすると編集できる。何らかの理由でアクシデントの項目を削除する場合は、選択後、下の × をクリックする。

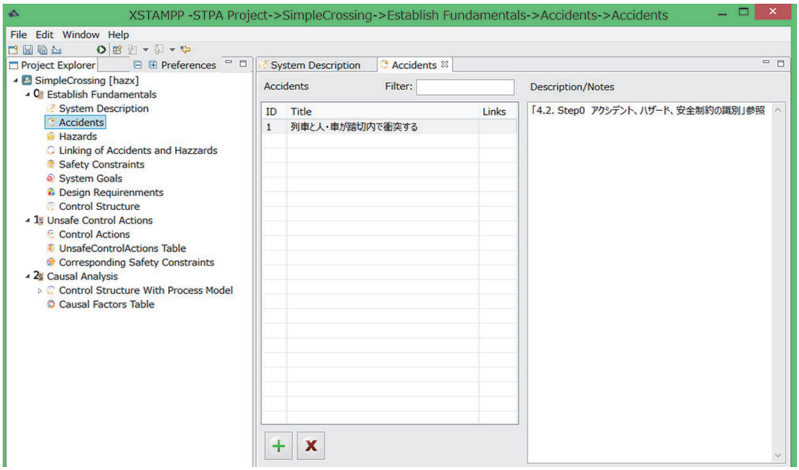


図 5.1-6　Accidents ステップ

"Hazards" も上記 "Accidents" とほぼ同様に記入、編集できる。図 5.1-7 はハザードを、「列車が在線中に踏切が閉まらない」、「踏切遮断後、列車が在線中に踏切が開く」の二つとして記入したものである。

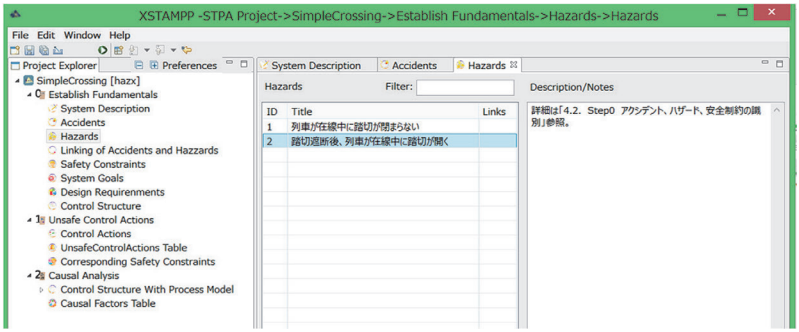


図 5.1-7　Hazards ステップ

"Linking of Accidents and Hazards" では、アクシデントとハザードを対応付ける。アクシデントとハザードが設定済みだと図 5.1-8 のように、アクシデントが左側に、アクシデントに対応付けるハザードが右上部に列挙される。あるアクシデント項目を選択した状態で、対応付けたいハザード項目を選択し、右中央付近にある "Add" をクリックすると対応付けが行われる。対応付けを解除したい場合は "Remove" を使う。ハザード下部にある "Switch to Hazards" を押すと、ハザードに対して、アクシデントを対応付けるように視点が変更された画面になる。対応付けが終わった後は、"Accidents" や "Hazards" を表示させると "Links" 欄に対応する ID が自動的に入力された状態になっているはずである。

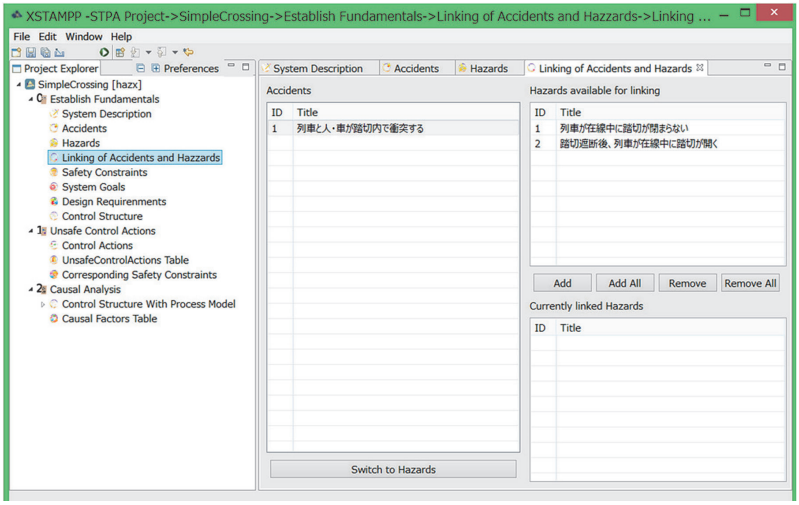


図 5.1-8　アクシデントとハザードの対応付け

コントロールストラクチャー（制御構造図）は、今回の分析では、まず図 5.1-9 のようなものとした。

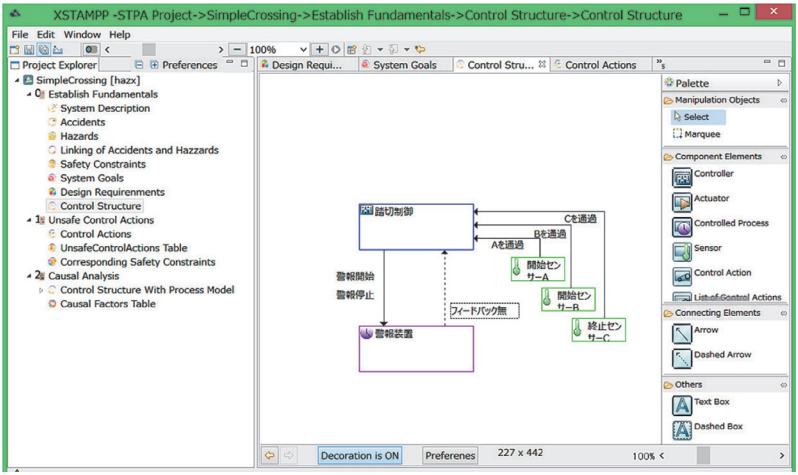


図 5.1-9　コントロールストラクチャー

1. Unsafe Control Actions

"Control Actions" の作業用のエリアを開くと、先の "Control Structure" で設定したコントロールアクションが自動的に入力済みの状態で表示される。今回の例では図 5.1-10 のようになる。この段階でもコントロールアクションを追加したり Title 編集したりできるが、更新が前のステップの成果物にさかのぼって反映されるとは限らない点に注意が必要である。

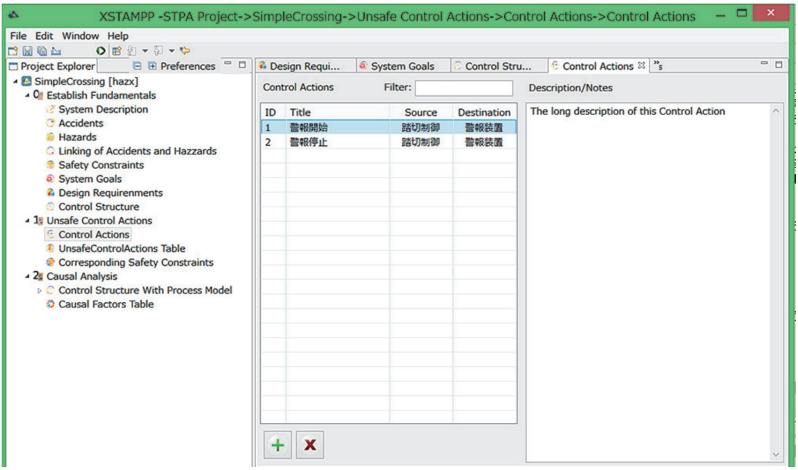


図 5.1-10　Control Action

“Unsafe Control Actions” を開くと、"Control Actions" で入力したコントロールアクションと、四つの非安全なコントロールアクションのパターンを使ったテンプレートの

表が表示される。 をクリックして非安全なコントロールアクション記述を追加する。また、 を使って対応するハザードとリンクさせる。（図 5.1-11 参照）

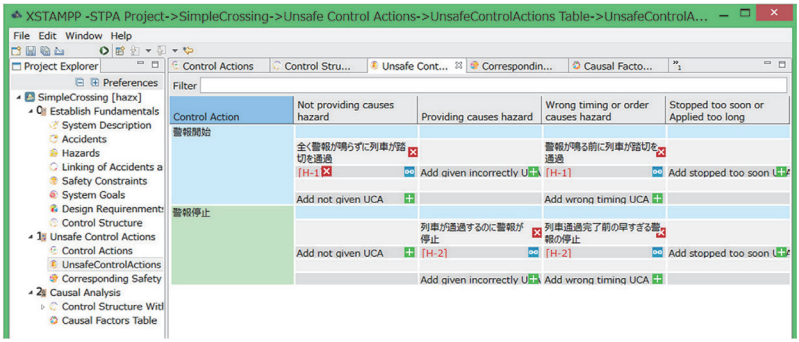


図 5.1-11　非安全なコントロールアクション

“2 Causal Analysis”

“Control Structure with Process Model” では、例えば図 5.1-12 のようにプロセスモデルを加えたコントロールストラクチャーを書いて分析を続ける。この分析は、典型的には、対象ドメインの専門家と協力して行い、分析とその結果に基づく修正を繰り返すことも多いとされている。

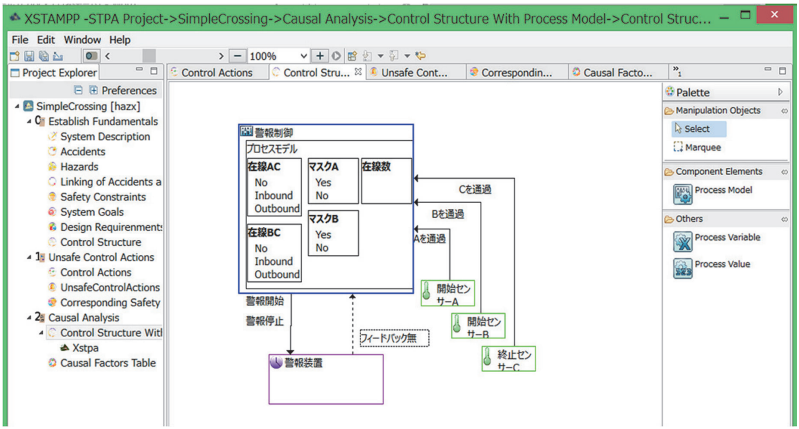


図 5.1-12　プロセスモデル

5.2. SysML 記述から STAMP 構築

本節では、SysML を利用した STAMP/STPA の事例を紹介する。今回の事例では、STAMP の構成要素のうち、1）アクシデント、ハザード、安全制約を要求図として記述し、2）コントロールストラクチャーを内部ブロック図として記述した。さらに、安全分析対象のコントロールアクション毎に Step1 と Step2 の分析結果を要求図として記述した。なお、今回の適用事例の詳細に関しては、「第 3 章対象システム概要」を参照されたい。

SysML 言語とそれを記述するツールを利用してコントロールストラクチャーを記述することで、以下に述べる効果が期待できる。コントロールストラクチャー構築時の課題としては、コントロールストラクチャーが複雑になると、人手による記述では手間がかかり過ぎ、間違いが混入しやすくなることが挙げられる。SysML 記述ツールを利用することで、コントロールストラクチャー記述に必要な手間を削減でき、コントロールストラクチャーの再利用性も向上する。（例えば、コンポーネント間の相互作用名やサブコンポーネントに対応するコントロールストラクチャーを一度記述すれば、それらのモデル要素が全体コントロールストラクチャー内に再度現れても簡単に複製・挿入できる。）また言語としての SysML を利用することで、SysML の各モデル要素の意味定義を利用できるため、記述したコントロールストラクチャーの曖昧さを低減できる。

アクシデント、ハザード、安全制約は表形式で記述されることが多い。表形式を用いることで、一覧性の高さや、他のツールとの親和性の高さといった利点が得られる。しかし、今回の事例では、これらを SysML の要求図として記述した。要求図として記述することで、あるアクシデントとあるハザードが対応する根拠を付記できるため、複数人による分析の際の相互理解が期待できる。さらに、表形式の場合には同じハザードや安全制約が複数個所に現れることがあり、ハザードや安全制約の内容を変更する際に、変更漏れが発生しやすくなる。他方、要求図ではそれらを一つにまとめられるため、変更漏れの低減が期待できる。また、ツールを用いることで、要求図から表形式へ容易に変換できる。

5.2.1. SysML 概説

システムズエンジニアリングは「システムを成功させるための複数の専門分野にまたがるアプローチと手段である（INCOSE, JCOSE 訳）」であると定義されている。[INCOSE] SysML（Systems Modeling Language）はシステムズエンジニアリングのためのモデリング言語である。SysML には大きく分けて要求図、構造図と振る舞い図の三種類の図が含まれ、さらに構造図はブロック定義図、内部ブロック図、パラメトリック図とパッケージ図に分けられ、振る舞い図はアクティビティ図、シーケンス図、状態機械図とユースケース図に分けられる。また、SysML 記述の構成要素や要素間の関係で理由が必要となる構成要素には、モデル要素「根拠」を用いて理由を付記できる。例えば要求図で、要求から要求を導出する場合、その導出関係に根拠を付記することで、その導出の妥当性を明記できる。また、根拠を明記することで、記述結果に至る過程を明確に残すことができる。[フリーデンタール 2012]

5.2.2. 安全制約の識別

この工程の入力は、安全分析の対象となる仕様や設計図等の資料であり、出力はアクシデント、ハザード及び安全制約を記述した要求図である。これらの入力やドメイン専門家へのインタビューから、アクシデントを初めに識別した。識別したアクシデントを基にハザード（アクシデントへ至る状態）を識別し、両者を導出関係で結び、識別根拠を導出関係に付記した。これらのハザードから安全制約を識別し、ハザードと安全制約

間を導出関係で結んだ。最後に、安全制約とそれを満たすべきシステム要素を充足関係で結んだ。

これらの作業結果、得られたアクシデント、ハザード、安全制約の一部を以下に列挙し、要求図を図 5.2-1　安全制約の導出（要求図）に示す。なお、アクシデントとハザードは要求ではなく、例えば、本来のアクシデントは「人命が失われる」のみである。要求として記述するために、本来のアクシデントとハザードは括弧（「」）で示し、文全体が要求なるように記述した。

- アクシデント：対象システムは「人命が失われる」状態へ至ることはない
- ハザード：対象システムは「列車在線時に踏切内に人が存在する状態」へ至らない
- 安全制約：列車在線時には、踏切制御装置は遮断機等に対して警報開始命令を指示する

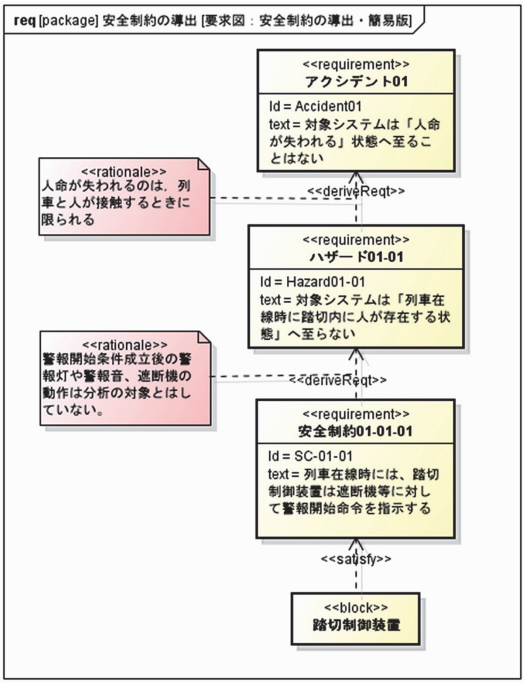


図 5.2-1　安全制約の導出（要求図）

5.2.3. コントロールストラクチャーの作成

この工程の入力は、安全分析の対象となる仕様や設計図等の資料であり、出力はコントロールストラクチャーを記述した内部ブロック図である。本事例の入力資料には、コントロールストラクチャーの構成要素が図示されていた。初めに、これらの構成要素を SysML のブロック図を利用して、物理的あるいは論理的観点から階層的にまとめた。次に、ブロック図から今回コントロールストラクチャーとして記述する構成要素の範囲を決め、内部ブロック図のひな型を構築した。最後に、入力資料等からコンポーネント間の情報のやり取りを抽出し、コンポーネント間の相互作用（コントロールアクション）

を追記した。なお、適切なブロック定義図と内部ブロック図を構築するには、何度か修正が必要であった。その際に、SysML ツールを利用することでモデル要素の再利用が容易になり、手作業と比較して、コントロールストラクチャー構築が容易であった。

これらの作業結果、得られたコントロールストラクチャーの内部ブロック図を図 5.2-2 に示す。なお今回の事例では、コントローラーを踏切制御装置とし、コントロール対象を「列車＋レール」として、各コンポーネントを見やすいように配置した。

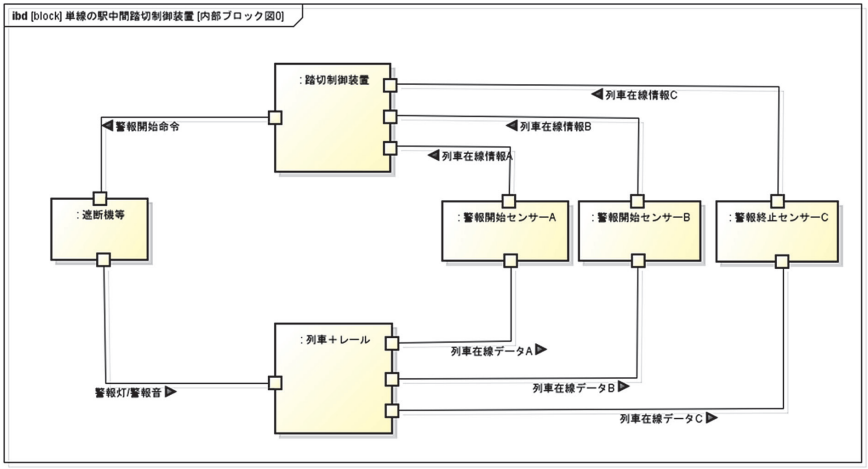


図 5.2-2　コントロールストラクチャー（内部ブロック図）

5.2.4. Step1 の実施

本事例では、分析対象のコントロールアクション毎に Step1,2 を実施した結果（要求図）を構築した。今回は踏切制御装置から遮断機等へのコントロールアクション「警報開始命令」に対する Step1,2 の実施結果のみを示す。

この工程の入力は、既に構築したアクシデント、ハザードと安全制約を記述した要求図及びコントロールストラクチャーを記述した内部ブロック図であり、出力は Step1 の分析結果を記述したコントロールアクション毎の要求図である。初めに、分析対象とするコントロールアクションを最上位ノードとする要求図を作成し、分析対象とするコントロールアクションに対し、Step1 の 4 つのガイドワードを適用した結果（非安全なコントロールアクションの候補）を、最上位ノード（コントロールアクション）の直下に記述し、両者を導出関係で結んだ。これらに対し Step1 を実施し、ハザードへ至る非安全なコントロールアクションに対しては、背景色を赤くして区別した。なお、1 つのガイドワードに対し、複数の非安全なコントロールアクションが対応することもある点には注意が必要である。Step1 の結果得られた要求図を図 5.2-3 に示すが、図 5.2-3 には Step2 の結果も合わせて記載されている点には注意が必要である。

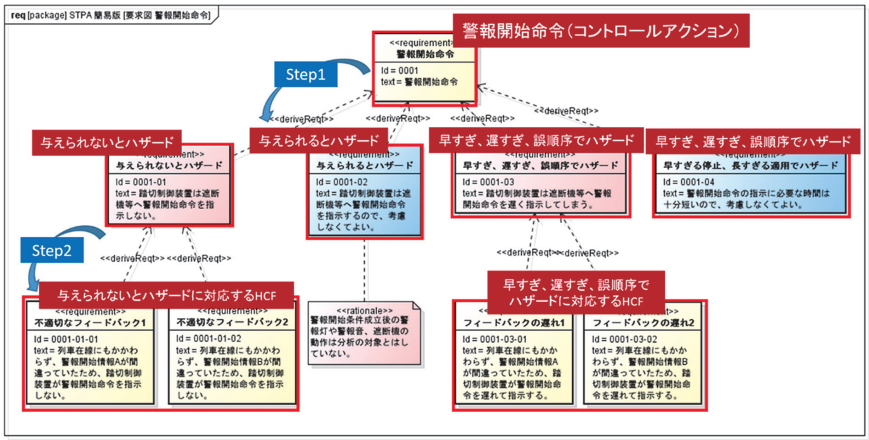


図 5.2-3　Step1,2 の結果（要求図）

5.2.5. Step2 の実施

この工程の入力は、は Step1 の分析結果を記述したコントロールアクション毎の要求図であり、出力はその要求図に Step2 の分析結果が追記された要求図である。初めに、Step1 で作成した要求図の非安全なコントロールアクションの直下に Step2 を実施して得られた hazard causal factor（HCF）を追記し、両者を導出関係で結んだ。このとき、ひとつの非安全なコントロールアクションに対して、複数の HCF が対応することや、逆にひとつの HCF に対し複数の非安全なコントロールアクションが対応することがもある点には注意が必要である。このような場合に、どの HCF が複数の非安全なコントロールアクションに対応しているかが、表形式と比較して対応関係が分かりやすくなっている。Step2 実施結果として得られる要求図を図 5.2-3 に示す。

5.2.6. まとめ

本節では、SysML を利用した STAMP 構築の事例を紹介した。今回紹介したの事例では、STAMP の構成要素のうち、1）アクシデント、ハザード、安全制約を要求図（図 5.2-1）として記述し、2）コントロールストラクチャーを内部ブロック図（図 5.2-2）として記述した。さらに、安全分析対象のコントロールアクション毎に Step1 と Step2 の分析結果を要求図（図 5.2-3）として記述した。

SysML 言語のモデル要素の定義を利用することで解釈の曖昧さを低減できた。また、SysML 記述ツールを利用したことで、特にコントロールストラクチャー内のコントロールアクションを一度定義すれば、コントロールアクションに対応するコンポーネントと合わせてツールが記憶するため、コントロールストラクチャーの修正が容易になり、作業の手間やミスの混入を削減できた。

5.3. 状態遷移図の活用

本節では、Step1 における UCA の識別に、状態遷移図を活用した例について述べる。

踏切において非安全な状態とは、列車が在線しているにもかかわらず、警報が鳴らない状態である。図 5.3-1 は、列車が踏切に到達した際（図 3.1-1 のモデルを考える）、A、C、B の各センサーが正常に検知したのか、検知できなかったのか（非検知）、あるいは誤検知をしてしまったのかで場合分けを行い、取り得る状態を網羅した状態遷移図である（列車が A から C に向かうときのみを考えている）。どのような動きから非安全な状態に至るのかを把握するためには、状態を漏れなく網羅することが重要となる。

図 5.3-1 からは、「非警報（非在線）」の状態のとき、A センサーが非検知だった場合には、直ちに非安全な状態に陥ることが読み取れる。また、「警報（在線）」の状態のときに、C センサーの誤検知があった場合も、同様に非安全な状態となる（図中の UCA1、UCA2、UCA3 は、前述の表 4.4-2 の UCA と対応している）。

Step1 では、UCA の識別を行うことになるが、どのような状態を取り得るのかを視覚的に整理し、表を作成する際の漏れをなくすためにも、状態遷移図を利用することは有効であると言える。

ただし今回、状態遷移図はセンサーの検知／非検知、及び誤検知にのみ着目しているが、STAMP では本来ならば、それ以外の要因も考慮しなければならない。例えば「列車が戻る」のような事象を表すためには表の方が書き易く、状態遷移図には表しづらいが、UCA 識別のための表作成の際に、状態遷移図を補完的に活用することによって、網羅的な分析がし易くなる場合もあるだろう。

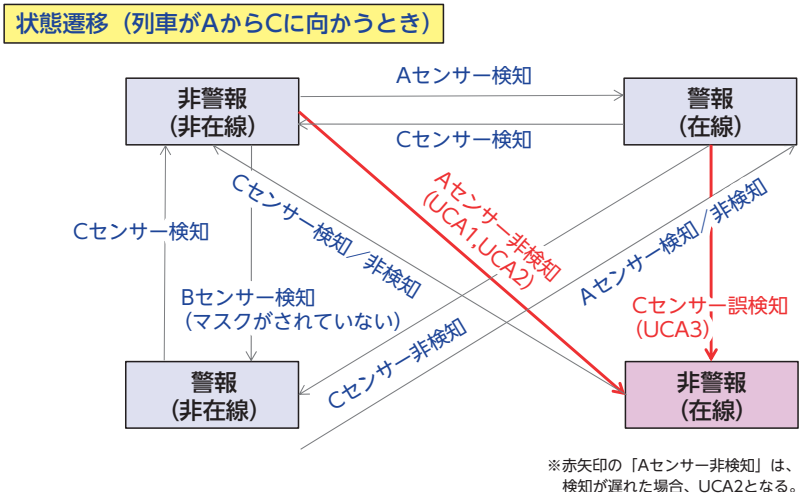


図 5.3-1　状態遷移図

5.4. 状態変数を用いた状況分析

第 2 章と 4 章で述べた標準的な STPA 分析においては、コントロールストラクチャー図を作成し、ガイドワードを適用して、非安全なコントロールアクションを識別する。これはシステムに関する概念的な情報だけで分析を可能とする利点があるが、識別はアドホックにならざるをえず、より系統的に、網羅的に分析するためには、それに適する手法が望まれる。その一つとして、コントロールアクションが与えられる状況（context）を系統的に定義し、非安全になるコントロールアクションを網羅的に識別する手法が提案されている [1]。その有効性を確かめるために、システムの状態変数を活用して状況を定義し、非安全なコントロールアクションの識別を試みたので、その概要を紹介する。

対象とするシステムは、第 3 章で述べた「単線の駅中間踏切制御システム」である。標準的な STPA 手順に従ってステップ 0 は実施する。その結果は次のとおりとする：

アクシデント：踏切横断中の人又は車両と列車が衝突する
ハザード：列車が在線中にもかかわらず、警報器が鳴動していない
安全制約：列車が在線中には、警報器を必ず鳴動させる
コントロールストラクチャー図：図 5.4-1 参照

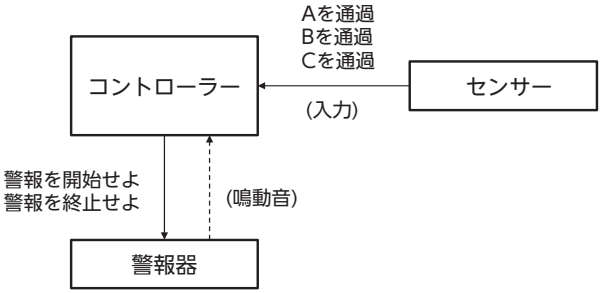


図 5.4-1　コントロールストラクチャー図

次のステップ 1 では、標準的なガイドワードを使うことによって、次の 4 通りの非安全なコントロールアクションを識別できる：

UCA1：列車が在線しているのに、警報開始が行われない
UCA2：列車が在線しているのに、警報開始が遅れて行われる
UCA3：列車が在線しているのに、警報終止が行われる
UCA4：列車が在線しているのに、警報終止が早く行われる

この段階での分析では、識別できる非安全なコントロールアクションは、安全制約のほぼ裏返しになっている。より詳細な分析を進めるために、システムの状態に関連する変数を導入し、それを用いてシステムの状況を定義し、それぞれの状況ごとに分析することを試みた。

対象システムには、「駅中間には列車は高々 1 台、又は同じ方向に走行している複数台が存在する」という前提条件がある。この条件を考慮して、状態変数として先行車と

[1] John Thomas 氏の 2013 年の博士論文を参考に。

後続車（簡単のために、後続車は高々 1 台とする）の位置を取り上げることにした。さらに、線路は、2 つの警報開始センサーと 1 つの警報終止センサーで区分されていると考えると、列車の位置は、次の 4 通りとなる（図 5.4-2 参照）：

位置 1：進行方向入口の警報開始センサーを通過する手前
位置 2：警報開始センサーを通過し、警報終止センサーを通過する手前
位置 3：警報終止センサーを通過し、出口の警報開始センサーを通過する手前
位置 4：出口の警報開始センサーを通過した後

したがって、先行車と後続車の位置関係は 16 通りとなる。それに、後続車は存在しない場合を加えると、状況数は、合計して 20 通りになる。

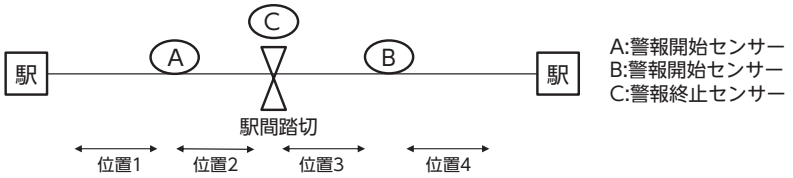


図 5.4-2　列車の位置

状態変数の組合せによって状況を表すと、一般的には状況数は多くなるので、それぞれの状況にガイドワードを適用して分析する前に、状況の絞り込みを行う必要がある。今回のケースでは、後続車は先行車の後ろという制約があるから、これを考慮すると、有意な組合せは表 5.4-1 に示す 9 通りになる。さらに列車が在線している状況だけがハザードに関係するから、分析すべき状況は 4 通りに減る。

表 5.4-1　列車位置による状況の分類

		列車の位置(X:先行車、Y:後続車) AとBは警報開始センサー、Cは警報終止センサー							
			A		C		B		
状況1	XA	X							列車数 1
2	AXC			X					
3	CXB					X			
4	YXA	YX							列車数 2
5	AYXC			YX					
6	CYXB					YX			
7	YAXC	Y		X					
8	YACXB	Y				X			
9	AYCXB			Y		X			

Aから進入したと仮定。有意な組合せは9通り。

在線状態

先行車がBを通過した状況は状況1、2、3に同じ

次に、列車の位置関係をもとに分類できた状況に対して、ガイドワードを適用してコントロールアクションがハザードを引き起こすかどうかを検討する。その結果の一部と

して、「警報を終止せよ」というコントロールアクションに対する結果を表 5.4-2 に示す。状況 2 は、先行車が警報開始センサーを通過し、警報終止センサーを通過する手前にいる状況であり、この表を見ると、状況 2 の結果は、標準的な分析で得られた UCA3 と UCA4 に相当し、標準的な分析と差があまりない。しかし、状況 9 は、先行車が警報終止センサーを通過し、出口の警報開始センサーを通過する手前にいて、後続車が入口の警報開始センサーを通過し、警報終止センサーを通過する手前にいる状況に対応し、状況 9 の結果は、2 台の列車の位置関係を反映し、標準的な分析では識別しづらい結果を導き出していることがわかる。

表 5.4-2　警報終止に対する状況分析

状況	制御行動	警報を終止せよ
		与えられるとハザード
状況2	AXC	1) Cを通過したと判断すると、警報が終止され、ハザードになる。 2) Cを通過し終える前に警報が終止されると、ハザードになる。
5	AYXC	状況2に同じ
7	YAXC	状況2に同じ
9	AYCXB	先行車がCを通過すると、後続車が在線しているにもかかわらず、警報が終止され、ハザードになる。

この後にステップ 2 を実施することになるが、これは標準的な分析と同じであり、ここでは省略する。

以上、2 台の列車の位置関係から状況を分析し、それぞれの状況で非安全なコントロールアクションを識別する手法の適用例を紹介した。標準的なガイドワードだけを使う分析に比べて、詳細で、網羅的な分析が行えるようになる。反面、状態変数が増えてくると、状況数が増大し、分析の手間が増えてしまう。適用に当たっては、組合せテスト件数を減らすことができる直交表などを使って、状況数を絞り込む工夫が必要になる。

6. エンタープライズ系システムでの STAMP 適用

STAMP では、事故は単純なコンポーネント故障のみを原因とせず、コンポーネントの振る舞いやコンポーネント間の相互干渉が、システムの安全性制約（コンポーネントの振る舞いに関わる物理的、人、又は社会に関わる制約）を違反した場合に起こると考える。

ここではエンタープライズ系システムにおいて、**アクシデントは IT サービス業務運用におけるサービス継続不可となる事象**と設定し STAMP/STPA の分析手法を適用によりそれを引起す要因を摘出し設計または改善に適用できる可能性できる可能性を検討する。

6.1. エンタープライズ系システムの特徴

エンタープライズ系システムは一般に情報系システムといわれているように、入力情報を集約・加工し、データベースに蓄積する業務、データベースから情報を検索・集約・加工し必要な情報を出力するシステムである。

エンタープライズ系システムは銀行・証券・クレジットカードなどの金融システム、各種手続きのための行政情報システム、電話・メール・放送等の情報通信システム、道路・鉄道・航空等の交通制御システムなど国民の安全な生活を支えるインフラシステムとなっており、システム障害等によるサービスの停止が発生した場合その影響は非常に大きい。

エンタープライズ系システムの構成はオンラインクライアント端末とそれを利用する人間、システムと人間をインタフェースする入出力情報からなり、ネットワークを含むサーバシステムはブラックボックスとして動作する。入出力情報には現金や IC カードなども対象となるが、ソフトウェアを組み込んだ携帯型端末や目的別専用入出力機器もある。

また、当該システムと外部システムとで情報授受を行う形態も存在する。

エンタープライズ系システムの特徴はデータ発生元の入力をトリガとし、構成機器を順次データが引き渡され、目的とする業務処理が実行されその結果が入力元に戻ってくる一連の同期処理となることである。これらと非同期に実施される処理には、運用業務、維持保守業務やシステムが状況に応じて自動的に動作環境を最適化させる機能などがある。

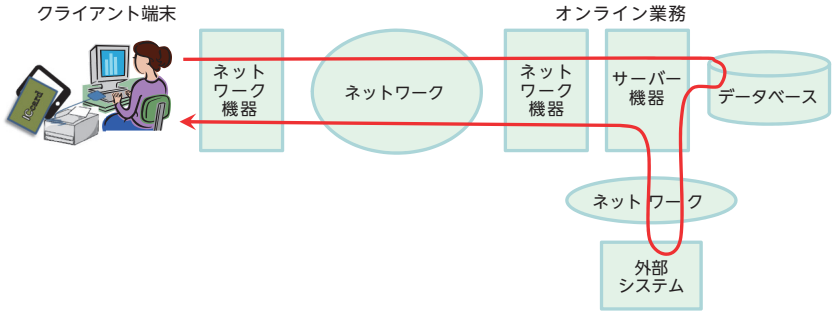


図 6.1-1　エンタープライズ系システムの同期処理の流れ

エンタープライズ系システムは基本的に運用管理システムのコントロールで自動運用され、必要に応じて保守部門のエンジニアは保守ツールを操作してシステム変更作業を実施する。

利用者はエンタープライズ系システムを利用してデータ投入あるいはアウトプットを受け取る。

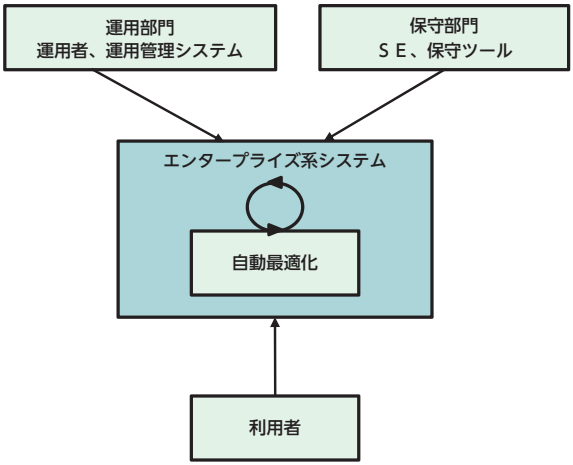


図 6.1-2　エンタープライズ系システムのコントロール

6.2. エンタープライズ系システムへの STAMP の適用について

エンタープライズ系システムがデータの一貫性、完全性を維持しつつシステム全体が遅滞無くサービスを継続することを STAMP での安全性の定義とすることで、この安全性を脅かす要因を抽出し、対策をするという局面で利用は可能と考える。

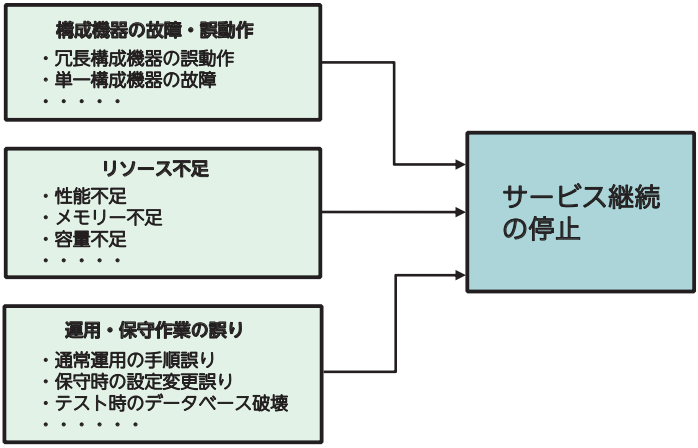


図 6.2-1　サービス継続を脅かす要因

それぞれの要因について、システム構築の上流工程の段階で STAMP/STPA 分析を行い、必要十分性を確認する。

STAMP/STPA の手順*に従って、以下の手順をひとつひとつ実施し、ハザードを洗い上げ、コスト対効果を評価しつつ極力漏れのない未然防止対策をすることになる。例えば、コントロールストラクチャーは運用部門の組織・人の指示・確認・承認のコミュニケーション構造をモデル化し運用操作ミスなどが発生しないような運用規定・ルール策定などに適用する。

＊（STAMP/STPA の手順）

①アクシデント、ハザード、安全制約の識別
②コントロールストラクチャーのモデル構築
③非安全なコントロールアクションの抽出
④非安全なコントロールの原因の特定

7. まとめ

本稿では、コンピュータソフトウェアによる制御、人間との協調制御、インターネットを介した制御などでますます複雑化しつつあるシステムの安全性解析を可能にする新しい手法 STAMP/STPA の理解・普及を目的にした WG の活動成果をまとめた。米国や欧州では、定期的なワークショップが何度か開催され、STAMP の有効性が実例に基づいて議論されているが、日本の中では、まだ十分な議論がされているとはいえない。WG では、こういった状況下で STAMP/STPA の理解と普及を促進するために、手法の長所や限界を教科書から抽象的に学ぶだけではなく、具体的な事例を用いて評価してゆくことを心がけてきた。しかしながら、安全に関わる生々しい事例では、関わった企業のノウハウや責任問題が絡んで正直な議論ができない可能性があること、逆に、単純すぎる事例では、本来の STAMP の価値を見逃す恐れがあること、の両方を考慮して事例を選ぶ必要があった。幸いにして、WG の中に、鉄道や宇宙分野の専門家、ならびに STAMP 解析の専門家を迎えることができ、その経験に基づいて、適切な STAMP 学習事例として、踏切制御の安全設計という題材を選ぶことができた。今回の報告では、単線踏切の制御という、踏切の中で簡単な事例をもとに、STAMP による安全評価を行ったが、次の視点で価値のある結果が得られたといえる。

(1) 踏切に関するドメイン知識を持っていない WG 委員により安全解析を行い、ドメイン知識を持った専門家と対等に安全設計に関する議論を行う。
(2) 踏切に関するハザードとその誘発要因を、体系的、かつ、漏れのないよう抽出し、その根拠を明確に表現する。

これらの成果は、STAMP の価値やその使い方を具体的に示す上で非常に有用であった。ここで得られたハザード誘発要因は、踏切の専門家からすれば、必ずしも新しいものではなく、経験的に知られ、その対策も行われているものではあるが、ドメイン専門家と非専門家が対等に安全設計を議論できることを実証したことは、将来の安全設計の第三者検証や可視化といった視点からも重要な成果である。踏切の安全設計に関しては、今回のような単純な事例だけでなく、運転管理、保守作業管理など、組織と人間の相互作用、コンピュータによる情報管理などが深く関わっており、STAMP の検証という視点から役に立つ事例がまだまだあると考えられる。今後も、いろいろな事例に関しての検討を継続してゆくことが望まれる。

また、このような安全システムの事例以外に、エンタープライズ系の損失防止（サービス提供の停止事故など）に関しても、STAMP の利用可能性について考察した。STAMP は、物理的な事故防止だけでなく、セキュリティ事故、エンタープライズ系のトラブル、サプライチェーンマネージメントでのトラブルなどの防止にも役立つと言われており、その応用可能性を今後も具体的に検討してゆく予定である。

一方、本稿では、上記の事例検証の他に、STAMP 解析支援ツール、既存のソフトウェアツール（SysML など）との組み合わせ方法などについても具体例を記載した。事例とツールの組み合わせは、STAMP の普及に欠かせないものである。

はじめての STAMP/STPA と題してまとめた報告書であり、事例が単純すぎるという批判もあるかもしれないが、本書を入門として利用頂き、本物の安全設計に役立てるきっかけにして頂ければ幸いである。

おわりに

ハザードの発生原因として「ソフトウェアの要求・設計ミス」を識別することは、長らく意味のないことと考えられていた。最大の理由は、設計ミスが無限のバリエーションを持ってしまうことであった。無限の原因を全て除去することはできない。それゆえ、ソフトウェアの信頼度は、便宜上、"100%" として計算に供され、ソフトウェアが原因で起こる事故は、適切な開発管理によって完全に除去できるという考え方がとられていた。

しかし、信頼度 100% という前提は明らかにおかしい。ソフトウェアのせいでシステムが意図しない挙動をすることは誰でも知っている。巨額の開発費用をかけて検証された OS に何万か所ものバグがあることも我々は知っている。しかし、ソフトウェアの安全工学は、長らく、信頼性工学と同一とみなされてきたばかりか、信頼度 100% という、間違った信頼性の数値を使いつづけるということになっていた。

そこに彗星のごとく登場したのが、本書で紹介している STAMP 手法を開発した Leveson 教授である。教授は、1995 年に、著書 "SAFEWARE" の中で、「ソフトウェアの信頼性が上がっても安全性は上がらない」と断言して我々の度肝を抜いた。1996 年に教授のサマーセミナーに出席した時、「むしろ信頼性が上がると安全性は下がるものがある」とまでの発言も飛び出し、出張報告書にそれを書いて提出した時、上司が烈火のごとく反論してきたことを思い出すのである。

教授の基本思想は当時から現在まで一貫している：

ソフトウェアが原因で起こる事故は、要求自体が間違っているからなのであって、故障とは関係が無い

"SAFEWARE" の中では、信頼性一辺倒から脱却せよと提言されていたが、無限のバリエーションを持つ要求ミスの排除方法の提示までには至っていなかった。STAMP は、この「無限」への回答なのである。無限をどのように有限化したのか。STPA Step1 の中で使用されている４つのガイドワードがそれである：

- 与えられないとハザード
- 与えられるとハザード（間違った条件）
- 早すぎ、遅すぎ、誤順序でハザード（間違った開始タイミング）
- 早すぎる停止、長すぎる適用でハザード（間違った終了タイミング）

アリストテレスが、神羅万象を二分分岐法によってカバーしうると主張したように、これらのガイドワードも、二分法によって巧妙に神羅万象を網羅している：

- 与えられる／与えられない　に分岐
- 正しい条件で与えられる／間違った条件で与えられる　に分岐
- 正しい条件で正しい開始タイミングで与えられる／間違った開始タイミングで与えられる　に分岐
- 正しい条件で正しい開始・終了タイミングで与えられる／間違った終了タイミングで与えられる　に分岐

STAMP（System Theoretic **Accident Model** and Process）の語源となった「事故モデル」とは、事故が発生するメカニズムというべきものであるが、STAMP の事故モデルは、事故を防止するための防御機能が提供される・されないということによって事故は発生するというものである。このパースペクティブにおいて、上記の二分分岐によるガイドワードは、事故発生のメカニズムを完全網羅する。ソフトウェア要求の間違い方は無限

にあるが、この事故モデルを採用することによって、有限化（しかもたったの４つ！）することができるというのが、STAMP のもたらした革命である。

本書の中で扱っている鉄道の踏切制御の評価においても、この４つのガイドワードを使い、網羅性を持った分析を実現できている。しかも、この４つのガイドワードを使うステップ（Step1）では、機能の詳細に立ち入ることもなしに、UCA を抽出することに成功した。すなわち、システム開発の最上流でソフトウェアのハザード解析が可能になったのである。さらに、ここからの Step2 において抽出されたハザード発生シナリオのうち、80% が故障以外の要因によるシナリオである。これらの成果は、従来の技術では全く達成不可能だった。ソフトウェアの仕様書なしにソフトウェアの安全解析を行うこと、故障に関わらないハザード発生シナリオを識別することで、従来不可能と考えられてきた、「ソフトウェアの要求・設計ミスによるハザード発生要因を識別する方法」を、ついに我々は手に入れたと言えるのではないだろうか。

このスキームは、今後どのように発展させることができるだろうか。STAMP の事故モデルは多分に防衛的である。システムの安全のためには、安全を守るための制御機能があるという前提に立っている。しかし、そのように安全機能が隅々まで安全を守っているシステムの開発が今後も主流であり続けるのかどうかは不明である。あらゆるものが相互接続してしまう IoT（Internet of Things）の時代においては、もはや全体制御機能は無く、超分散・無政府主義的な巨大システムの方がむしろ一般的となるのではないだろうか。STAMP は、事故は故障からではなく、コンポーネント間のインタラクションから「創発的に」発生すると説いている。明確な「故障」によらず、インタラクションから創発的に発生するハザード要因を見つけ出す手法は確立された。次の課題は、その最も極端な形態である超分散システムで発生する事故かもしれない。そのようなシステムの事故モデルを考えなければならない時代に、既に我々は生きているのである。

2016 年 12 月 5、6 日に九州大学にて、日本ではじめての STAMP ワークショップが計画されているが、多くの産業界からの参加を期待したい。

URL：https://sites.google.com/site/stampwsj/

参考文献

第 1 章

[1] [Leveson2012]
Nancy G. Leveson:Engineering a Safer World: Systems Thinking Applied to Safety（Engineering Systems）, The MIT Press, 2012.

第 2 章

[2] [STPA2015]
An STPA Primer、http://psas.scripts.mit.edu/home/wp-content/uploads/2015/06/STPA-Primer-v1.pdf

[3] [IPA2015]
STAMP 手法に関する調査報告書、http://www.ipa.go.jp/sec/reports/20150918.html

[4] [Leveson2011]
Nancy Leveson. 16.863J System Safety, Spring 2011.（Massachusetts Institute of Technology: MIT OpenCourseWare）, http://ocw.mit.edu（Accessed 6 Jan, 2016）.

第 5 章

[5] [INCOSE]
What is Systems Engineering? , INCOSE, http://www.incose.org/AboutSE/WhatIsSE

[6] [フリーデンタール 2012]
フリーデンタール、ムーア、スタイナー : システムズモデリング言語 SysML、東京電機大学出版局、2012

索引

付録

A）用語説明

Accident
　望ましくない事象、事故・損失。アクシデント。

Hazard
　アクシデントが潜在している具体的な状態。ハザード。

UCA（Unsafe Control Action）
　非安全制御行動。事故・損失（アクシデント）につながる制御、制御行動、動作。

HCF（Hazard Causal Factor）
　ハザード誘発要因。危険な状態（ハザード）を引き起こす原因。

コントロールアクション（Control Action）
　コントローラーが被コントロールプロセスに対して行なう制御指示・制御動作・制御行動。

コントロールストラクチャー（Control Structure Diagram）
　制御構造図。システムにおいて、安全制約の実現に関係するコンポーネント、およびコンポーネント間の相互作用から成る構造図。

コントロールループ（Control Loop）
　コントローラー、被コントロールプロセス、コントロールアクション、フィードバックから成る循環関係。

FMEA（Failure Mode and Effect Analysis）
故障モード影響解析。ボトムアップで故障原因がシステム全体に及ぼす影響を解析する方法。対象とするコンポーネントの故障モードをリストアップし、故障原因・システムへの影響評価・影響の深刻度を表にまとめる方法。

FTA（Fault Tree Analysis）
故障木解析。故障の原因をトップダウンで分析する方法。フォールトツリー図や影響分析表を用いてハザード要因を分析する方法。機器や組織の単一故障をハザード要因として識別し、分岐条件を論理的に組むことで網羅的に分析できる。

HAZOP（HAZard and OPerability study）
サブシステム・コンポーネント間の複雑な干渉があるシステムのハザード分析法。7つのガイドワードを用いて設計意図からの逸脱によるハザードを洗い出す手法。

プロセスモデル
コントローラーが想定する被コントロールプロセスの状態。

4 種類のガイドワード
非安全コントロールアクション（UCA）の分類であって、UCA を抽出する際のヒントとなる、次の 4 つの言葉を指す。

◈（**与えられないとハザード：Not Providing**）安全のために必要とされるコントロールアクションが与えられないことがハザードにつながる。

◈（**与えられるとハザード：Providing causes hazard**）ハザードにつながる非安全なコントロールアクションが与えられる。

◈（**早過ぎ、遅過ぎ、誤順序でハザード：Too early/too late, wrong order causes hazard**）安全のためのものであり得るコントロールアクションが、早すぎて、遅すぎて、または順序通りに与えられないことでハザードにつながる。

◈（**早過ぎる停止、長過ぎる適用でハザード：Stopping too soon/applying too long causes hazard**）（連続的、または非離散的なコントロールアクションにおいて）安全のためのコントロールアクションの停止が早すぎる、もしくは適用が長すぎることがハザードにつながる。

編著者（敬称略）

システム安全性解析手法 WG（主査、以下 50 音順）

主査	荒木 啓二郎	国立大学法人九州大学
	大原 衛	地方独立行政法人東京都立産業技術研究センター
	岡本 圭史	独立行政法人国立高等専門学校機構 仙台高等専門学校
	兼本 茂	公立大学法人会津大学／障害原因診断 WG 主査
	川野 卓	東日本旅客鉄道株式会社
	日下部 茂	国立大学法人九州大学
	中村 洋	株式会社レンタコーチ
	野本 秀樹	有人宇宙システム株式会社
	福田 和人	東日本旅客鉄道株式会社
オブザーバー		
	星野 伸行	有人宇宙システム株式会社

IPA/SEC（50 音順）

石井 正悟
石田 茂
十山 圭介
松田 充弘
三縄 俊信
三原 幸博
八嶋 俊介

はじめての STAMP/STPA
～システム思考に基づく新しい安全性解析手法～

2016 年 4 月　第 1 刷発行

発行　独立行政法人 情報処理推進機構（IPA）

〒 113-6591 東京都文京区本駒込 2-28-8
文京グリーンコート　センターオフィス 16 階

制作　株式会社サンワ　Printed in Japan

IPA 独立行政法人情報処理推進機構
Information-technology Promotion Agency, Japan

技術本部 ソフトウェア高信頼化センター
Software Reliability Enhancement Center(SEC)